KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT ECONOMISCHE EN
TOEGEPASTE ECONOMISCHE WETENSCHAPPEN
DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

DEVELOPING INTELLIGENT SYSTEMS FOR CREDIT SCORING USING MACHINE LEARNING TECHNIQUES

Proefschrift voorgedragen tot
het behalen van de graad
van Doctor in de Toegepaste
Economische Wetenschappen
door
Bart BAESENS

Nummer 180 2003

Committee

Acknowledgments

During my period as a researcher, I have had the pleasure of collaborating with many interesting and fine people whose contribution and support improved the quality of my research. Therefore, I would like to start by expressing my gratitude and acknowledgements.

First of all, I would like to thank my promotor, prof. dr. Jan Vanthienen, for the very pleasant collaboration and for the freedom he granted me to pursue my research goals. During my career as a PhD research assistant, he was always prepared to give me advice and support (even when his schedule was very busy), and the many discussions we had were both very interesting and stimulating. Professor Vanthienen taught me that data mining is not only about complex algorithms and numbers but should always be approached with a sound sense of criticism. This is a principle which, in my opinion, is far too often neglected in today's academic as well as business practice. I also would like to thank the other members of my PhD committee for carefully reading my text and providing me with their comments. I am grateful to prof. dr. Jacques Vandenbulcke for triggering my interest in the field of management information systems and for the many opportunities and challenges he gave me. I thank prof. dr. Maurice Verhelst for the very nice collaboration and for giving me the opportunity to teach the exercises in statistics which certainly helped to improve my presentation skills. I am indebted to prof. dr. Martina Vandebroek for teaching me the basic concepts of multivariate statistics and classification which further aroused my interest in data analysis and data mining. My two external committee members, professor Jonathan Crook from the University of Edinburgh and professor Lyn Thomas from the University of Southampton, also deserve a special acknowledgment for their constructive comments and for allowing me to participate in the Credit Scoring and Credit Control conference.

I would also like to acknowledge the other professors - prof. dr. Guido Dedene, prof. dr. Ferdi Put, prof. dr. Monique Snoeck, prof. dr. Wilfried Lemahieu - and colleagues - Stijn Viaene, Jurgen Martens, Frank Goethals, Cindy Michiels, Herlinde Leemans, Dirk Vanderbist, Hilde Plessers and Stephan Poelmans - of the Management Informatics research group for the nice working atmosphere. I

want to deeply thank my office colleague and friend dr. Christophe (he doesn't like to be referred to as dr. Mues) for helping me to overcome the many acute outbursts of doctoratitis that tortured me during my period as a PhD researcher. Also, the expertise he passed on to me with respect to the basic principles of office soccer and thesis throwing, and the insights gained from the many fruitful political discussions (often ideally timed), are assets that I will cherish during the rest of my career. Manu De Backer insisted on thanking him for his invaluable advice on figure formatting, the many gastronomic evenings we shared accompanied by french fries, hamburgers, beer and Champions League football, and for allowing me to calm my anger and rage during the many exciting and heroic games of ping-pong we played (and which I dominantly won!).

I have had the pleasure of working with very interesting people both in a national and international context. I would like to thank dr. ir. Tony Van Gestel from Dexia Group for the joint work we did on using least squares support vector machines for practical application purposes. The numerous time we spent on debugging and running Matlab programs and the many discussions we had on maximizing our financial asset returns (or how to reduce our losses) were both interesting and (sometimes even) profitable. I am also grateful to his colleagues, prof. dr. ir. Johan Suykens, prof. dr. ir. Bart De Moor and prof. dr. ir. Joos Vandewalle for their support and feedback. I would also like to thank prof. dr. Rudy Setiono from the University of Singapore for the joint work we did on neural network rule extraction. I acknowledge prof. dr. Frank Hoffmann from the University of Dortmund for our collaboration on genetic fuzzy rule extraction. Although all my conversations with Rudy and Frank thus far took place electronically, I do hope to meet them in person one day. I am indebted to prof. dr. Dirk Van den Poel and prof. dr. Patrick Van Kenhove from the University of Ghent for our joint work on using machine learning techniques and data mining in a marketing context. Also Geert Verstraeten and Wouter Buckinx are acknowledged for the fruitful collaboration. I am grateful to dr. Michael Egmont-Petersen from the University of Utrecht for the joint work we did on Bayesian probabilistic networks. I would like to thank An Carbonez and Martine Beullens from the University Centre of Statistics for allowing me to assist in organising the data mining courses and for the software support. A special word of thanks goes to Annelies Bulkens and all other people from SAS Belgium for the pleasant collaboration, the software support, and for allowing me to participate in all kinds of very interesting (and well-organised!) SAS events. A further word of thanks goes to the following people who kindly provided me with data allowing me to conduct my research: Joseph Vingerhoets (Axa Bank), Rudi Stommels (Dexia Bank), dr. Joao Garcia (Dexia Group), Peter Van Dijcke (Dexia Group) and dr. Maria Stepanova (UBS Bank, Switzerland). I am also grateful to the master thesis students that assisted me in my research and to the editors of all journals for which I was asked to review the work of colleagues in the field. Chris Massie and the other members of the secretariat are acknowledged for guiding me through the administrative steps of the process.

Finally, I would like to thank my parents, brother and sister, parents-in-law, other family members, and friends for their support. Especially my wonderful wife, Katrien Denys, deserves a word of thanks for her continuous love and care. Her friendship and humor helped me to reset (or should I say set?) my brain when I needed it. I hope I can be as good a husband to you as you are a wife and partner to me!

Contents

1 Introduction **1**

1.1 The Knowledge Discovery in Data Process 2

1.2 The Credit Scoring Classification Problem 4

1.3 Contributions 5

1.3.1 Benchmarking state of the art classification algorithms . . . 5

1.3.2 Investigating the impact of various cut-off setting sche- mes on scorecards 5

1.3.3 Developing intelligent systems for credit scoring using neural network rule extraction and decision tables 6

1.3.4 Developing intelligent systems for credit scoring using fuzzy rule extraction 6

1.3.5 Using a neural network based approach for predicting customer default times 6

1.4 Notation 7

2 An Overview of Classification Techniques and Issues **9**

2.1 Classification Techniques 10

2.1.1 Logistic Regression 10

2.1.2 Discriminant Analysis 11

2.1.3 Linear Programming 14

2.1.4 Bayesian Networks for Classification 14

2.1.5 The naive Bayes classifier 17
2.1.6 Tree augmented naive Bayes classifiers 18
2.1.7 Decision Trees and Rules 19
2.1.8 K-Nearest Neighbour Classifiers 22
2.1.9 Neural Networks 22
2.1.10 Support Vector Machine Classifiers 27
2.1.11 Least Squares Support Vector Machine Classifiers 29
2.2 Data Set Split Up 30
2.3 Input Selection 32
2.4 Example Benchmarking Study 1 34
2.4.1 Description of the Data Sets 34
2.4.2 Hyperparameter Selection for the SVM Classifiers 35
2.4.3 Experimental Setup 36
2.4.4 Results 38
2.5 Benchmarking Study 2 40
2.6 Conclusions and Limitations of Benchmarking Studies 1 and 2 . . 43
2.7 The Area Under the Receiver Operating Characteristic Curve . . . 45
2.8 Computing the AUC 47
2.9 Test Statistics to Compare the PCC and AUC 48
2.10 Conclusions 50

3 Building Scorecards for Credit Scoring 53

3.1 Problem Statement 54
3.2 Reject Inference 56
3.3 Using Machine Learning Techniques for Credit Scoring 58
3.4 Building Scorecards for Credit Scoring 61
3.4.1 Data Sets and Experimental Setup 61

3.4.2 Setting the cut-off 62

3.4.3 Results 64

3.4.4 Discussion 66

3.5 Conclusions 73

4 Neural Network Rule Extraction for Credit Scoring 75

4.1 An Overview of Neural Network Rule Extraction 76

4.2 Neural Network Rule Extraction using Neurolinear and Neurorule 80

4.3 Neural Network Tree Extraction using Trepan 86

4.4 Neural Network Rule Extraction for Credit Scoring 88

4.4.1 Experimental Setup 88

4.4.2 Results for the Continuous Data Sets 89

4.4.3 Results for the Discretized Data Sets 92

4.5 Visualizing the Extracted Rule Sets using Decision Tables 98

4.6 Conclusions 106

5 Fuzzy Rule Extraction for Credit Scoring 109

5.1 Fuzzy Classification Rules 110

5.2 Using Fuzzy Rules for Credit Scoring 111

5.3 Evolutionary Algorithms for Inferring Fuzzy Rules 113

5.3.1 Evolutionary Algorithms 113

5.3.2 Boosted Genetic Fuzzy Classification 114

5.3.3 Genetic Representation for Approximate and Descriptive Fuzzy Rules 115

5.3.4 Fitness Function 116

5.3.5 The Boosting Algorithm 118

5.3.6 Fuzzy Classifier Aggregation 119

5.4 Neural Network Fuzzy Rule Extraction using Nefclass 120

5.5 Empirical Evaluation 123
5.5.1 Data sets and Experimental Setup 123
5.5.2 Results 124
5.6 Conclusions 130

6 Survival Analysis for Credit Scoring 131
6.1 Basic Survival Analysis Concepts 132
6.2 Statistical Methods for Survival Analysis 134
6.2.1 Kaplan Meier Analysis 134
6.2.2 Parametric Survival Analysis Models 134
6.2.3 Proportional Hazards Models 136
6.2.4 Discrete Proportional Hazards Models 139
6.3 Neural Networks for Survival Analysis 140
6.3.1 Direct Classification 141
6.3.2 Ohno-Machado 141
6.3.3 Ravdin and Clark 142
6.3.4 Biganzoli et al. 143
6.3.5 Lapuerta et al. 143
6.3.6 Faraggi 144
6.3.7 Street 144
6.3.8 Mani 145
6.3.9 Brown et al. 146
6.3.10 Discussion 146
6.4 Survival Analysis for Credit Scoring 146
6.5 Empirical Evaluation 149
6.5.1 Experimental Setup and Data Set Characteristics 149
6.5.2 Results for Predicting Early Repayment 153

6.5.3 Results for Predicting Default 155
6.6 Conclusions 162

7 Conclusions 163
7.1 Thesis Summary and Principal Conclusions 163
7.2 Issues for Further Research 166
7.2.1 The Knowledge Fusion problem 166
7.2.2 Extensions to Indirect Credit Scoring 167
7.2.3 Behavioral Credit Scoring 168
7.2.4 Extensions to other Contexts and Problem Domains 168

A Tables Accompanying Benchmarking Study 1 of Chapter 2 169

B Multiclass Benchmarking Study 171

C Attributes for German credit 175

D Attributes for Bene1 179

Samenvatting

Het ontwikkelen van intelligente systemen voor kredietoekenning met behulp van machine learning technieken

Financiële instellingen hebben gedurende de laatste decennia massa's gegevens verzameld aangaande het terugbetalingsgedrag van hun klanten[1,2]*. Samen met de toename in rekencapaciteit, geboekt op het vlak van hardware, en de opkomst van geavanceerde algoritmen creëert dit een nieuwe uitdaging: hoe kunnen we uit deze verzamelde gegevens nuttige beslissingsmodellen extraheren en deze succesvol aanwenden als hulpinstrument bij de kredietbeslissing voor toekomstige klanten.*

De extractie van kennis en beslissingsmodellen uit data, en het daarmee geassocieerde traject, wordt doorgaans onder de noemer Knowledge Discovery in Data (KDD) gevat. Machine learning is een essentieel onderdeel van KDD en voorziet in een ganse waaier van leeralgoritmen gericht op het extraheren van kennis en patronen uit data. In dit doctoraat gaan wij dieper in op het gebruik van machine learning technieken bij de ontwikkeling van beslissingsondersteunende systemen voor kredietverlening. Hierbij zullen zowel de accuraatheid als de begrijpbaarheid van de geëxtraheerde modellen een cruciale rol spelen.

[1]B. Baesens, C. Mues, J. Vanthienen, Knowledge Discovery in Data: van academische denkoefening naar bedrijfsrelevante praktijk, *Informatie*, pp. 30-35, Februari, 2003.

[2]B. Baesens, C. Mues, J. Vanthienen, Knowledge Discovery in Data: naar performante *én* begrijpelijke modellen van bedrijfsintelligentie, *Business IN-zicht*, Nummer 12, Maart 2003.

Business Intelligence en Knowledge Discovery in Data

Business Intelligence omsluit een brede categorie van ICT-applicaties en -technologieën voor het verzamelen, analyseren en verspreiden van bedrijfsinformatie, die de bedoeling hebben om de bedrijfsvoering te ondersteunen of te optimaliseren. Daarbij duiken doorgaans begrippen op als data warehousing, data mining en knowledge management. Met name het distilleren van bruikbare patronen uit de almaar groeiende stroom van ruwe data vormt een sleuteluitdaging binnen dit geheel. De geautomatiseerde ontginning van kennis, en het daarmee geassocieerde traject, wordt doorgaans onder de noemer Knowledge Discovery in Data (KDD) gevat. Het is een iteratief proces dat ruwweg uitgesplitst kan worden in drie fasen:

1. sampling en data preprocessing;
2. data mining;
3. de ontwikkeling van beslissingsondersteunende systemen.

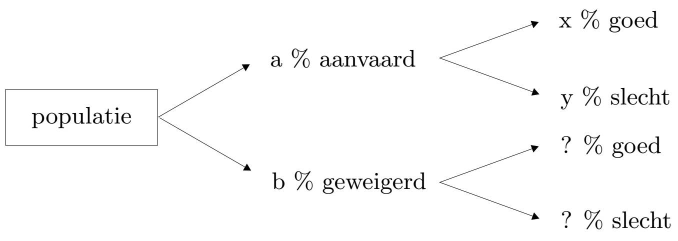
In de sampling en data preprocessing fase worden de relevante gegevensbronnen geïdentificeerd, de data geselecteerd en vervolgens opgeschoond. Vertrekkende van de aldus bekomen dataset wordt, in de daaropvolgende data mining fase, kennis geëxtraheerd door de uitvoering van een *machine learning* algoritme; het resultaat hiervan kan bijvoorbeeld de vorm aannemen van een neuraal netwerk, beslissingsboom, of een set van associatieregels. In de laatste fase wordt de geëxtraheerde kennis dan geïntegreerd in het betreffende bedrijfsproces via de ontwikkeling van een beslissingsondersteunend systeem.

In een kredietverleningscontext kan KDD toegepast worden voor de ontwikkeling van modellen die de kredietwaardigheid van toekomstige klanten voorspellen. Gebaseerd op de kenmerken en het terugbetalingsgedrag van klanten uit het verleden tracht men hierbij modellen te schatten die de kans op succesvolle terugbetaling van nieuwe potentiële klanten zo nauwkeurig mogelijk berekenen (ook wel credit scoring genoemd). Op basis hiervan kan dan een beslissing genomen worden om de kredietaanvraag te aanvaarden dan wel te verwerpen. Het spreekt vanzelf dat we hier met een klassiek binair classificatieprobleem te maken hebben: is de klant een wanbetaler of niet, gegeven zijn inkomen, spaarmiddelen, huwelijksstatus, etc.? Het gebruik van KDD voor het ontwikkelen van intelligente systemen voor kredietverlening vormt één van de kernonderzoeksvragen van dit doctoraat. Andere interessante toepassingen van KDD bestaan in bijna alle functionele domeinen waar voldoende data voorhanden zijn. Enkele frequente voorbeelden vinden we in marketing - we denken hierbij aan market basket analyse, waar het de bedoeling is patronen in het aankoopgedrag van klanten op te sporen, of bijvoorbeeld het voorspellen van klantverloop (churn prediction) -, in financiewezen (bijvoorbeeld stock picking), fraudedetectie, en zo meer.

In wat volgt worden de verschillende stappen van het KDD proces nader toegelicht en worden de bijdragen en onderzoeksvragen van dit doctoraat gesitueerd.

Sampling en Data preprocessing voor Credit Scoring

Hoewel het belang van een goed uitgevoerde preprocessing in de KDD literatuur al vaker is benadrukt, willen wij er nogmaals de aandacht op vestigen. Deze fase is immers cruciaal voor het succes van de volgende stappen in het KDD-proces. Onze ervaring heeft meermalen uitgewezen dat zich bij de uitvoering van een KDD-project al snel praktische vragen opdringen over sampling en preprocessing die vaak op een ad-hoc manier beantwoord (moeten) worden. In een kredietverleningscontext bijvoorbeeld blijkt de precieze definiëring van de klantenpopulatie allerminst een triviale opdracht. Immers, in het verleden voerden banken ook al een kredietverleningspolitiek, waardoor de klantenpopulatie eigenlijk uit twee subpopulaties bestaat: die aanvragers aan wie krediet werd toegekend en diegenen aan wie krediet geweigerd werd. Voor de eerste subpopulatie is bekend welke klanten uiteindelijk wanbetalers bleken en welke niet. Voor de tweede subpopulatie is dit echter niet bekend (zie Figuur I). Dat brengt uiteraard met zich mee dat de tweede subpopulatie bij gebrek aan waarde voor de afhankelijke variabele niet gebruikt kan worden bij het schatten van klantscoremodellen. Dit fenomeen, en



Figuur I: Het reject inference probleem.

het mogelijk vertekenend effect ervan op de resultaten, wordt in de credit scoring literatuur vaak omschreven als het reject inference probleem en is tot op heden vanuit academische hoek nog niet op een sluitende manier beantwoord. Vaak worden hier dan ook heuristische oplossingsmethoden voorgesteld en hanteert men het principe dat je moet roeien met de riemen die je hebt.

Een ander voorbeeld betreft market-basket analyse, waarbij men aan de hand van associatieregels patronen probeert te identificeren in de aankooptransacties van klanten. Ook hier dient grondig nagedacht te worden over hoe de aankooptransacties gesampled moeten worden vooraleer het data mining algoritme uitgevoerd kan worden. Een belangrijk aandachtspunt hierbij betreft de timing van de transacties:

associaties tussen bijvoorbeeld skibroeken en -brillen zijn doorgaans prominenter aanwezig tijdens de wintermaanden dan gedurende de rest van het jaar.

Ook de uitvoering van andere preprocessing taken, zoals bijvoorbeeld het opschonen van de data, de identificatie van extreme observaties (wat is een extreme observatie?), de adequate definitie van de doelvariabele (wat is een wanbetaler?) en de keuze van de juiste onafhankelijke variabelen roepen heel wat praktische overwegingen op en houden vaak trade-offs in waarmee niet onberedeneerd mag worden omgegaan. Samengevat is de sampling en preprocessing fase een cruciale stap binnen het KDD proces. In deze stap worden immers keuzes gemaakt die zowel vanuit academisch als bedrijfsperspectief te verantwoorden moeten zijn. Maar al te vaak vergeet men dat de kennis die ontgonnen wordt gedurende het verdere verloop van het KDD-proces altijd geïnterpreteerd en teruggekoppeld moet worden in functie van de keuzes die in de preprocessing stappen zijn gemaakt.

Machine learning technieken voor credit scoring

Zodra de data is gepreprocessed, kan de data mining fase worden aangevat. Naargelang de problematiek kan hierbij een onderscheid gemaakt worden tussen twee vormen van data mining: voorspellende en beschrijvende (zie Tabel I). Een prangend

Data mining	**Taak**	**Karakteristiek**	**Voorbeeld**
voorspellende data mining	regressie	voorspellen van een continue afhankelijke variabele	voorspellen van beurskoersen, productverkoop
	classificatie	voorspellen van een discrete afhankelijke variabele	voorspellen kredietwaardigheid, fraude, bankroet
beschrijvende data mining	clustering	identificeren van homogene subpopulaties	identificeren van marktsegmenten
	associatieregels	zoeken van verbanden tussen items	identificeren van producten die samen worden aangekocht (market basket analysis)
	sequentieregels	zoeken van verbanden tussen items in de tijd	identificeren van tijdsvolgorde van aankoop producten
	afhankelijkheidsanalyse	identificeren van afhankelijkheden tussen variabelen	identificeren van afhankelijkheden tussen biomedische metingen

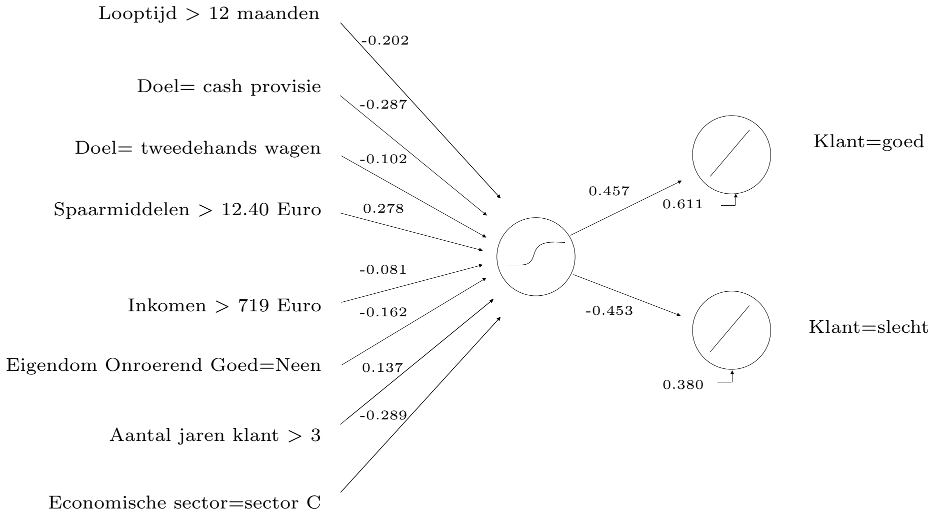
Tabel I: Voorspellende versus beschrijvende data mining.

probleem dat zich hier stelt is de keuze van een geschikt algoritme. Alleen al voor classificatietoepassingen zoals kredietoekenning wordt het KDD-project team onvermijdelijk geconfronteerd met een brede waaier aan algoritmen: statistische classificatiemethoden, lineaire programmering, neurale netwerken, beslissingsbomen, k-nearest neighbour, Bayesiaanse netwerken, genetische algoritmen, en ga zo maar verder [13, 242]. Een recent ontwikkelde methode van classificatie is zelfs gebaseerd op het modelleren van het gedrag van mierenkolonies. Heel wat KDD-tools (SAS Enterprise Miner, SPSS Clementine en IBM Intelligent Miner for Data) bieden ondersteuning voor deze methoden. Bij de keuze van een geschikt algoritme moeten echter meerdere karakteristieken in ogenschouw genomen worden.

Hoewel nauwkeurigheid een intuïtief prestatiecriterium lijkt, moet gezegd worden dat een ondubbelzinnige kwantificering ervan niet altijd voor de hand ligt. Denk bijvoorbeeld aan de kredietcontext. Als eerste naïeve benadering kan men streven naar het maximaliseren van het aantal correct geclassificeerde klanten op een onafhankelijke testset. Men kiest hierbij vooraf een cut-off (standaard 0.5), zet de posterior kansen P(klant=goede terugbetaler | inkomen, spaarmiddelen, huwelijksstatus, ...) die de classificatie-techniek oplevert om naar klasse-labels (goede of slechte terugbetaler), en vervolgens berekent men dan de procentuele accuraatheid op de testset. Hoewel dit geen slecht criterium is, vertoont het toch een aantal tekortkomingen. Slechts een klein aantal klanten zal wanbetaler zijn, wat ervoor zorgt dat een weinig informatieve regel zoals 'elke klant is een goede klant' reeds een heel goede prestatie oplevert. Men moet, met andere woorden, dus ook andere aspecten beschouwen, zoals de misclassificatiekosten van vals negatieven versus die van vals positieven. Deze zijn echter moeilijk te kwantificeren, aangezien de kosten typisch zullen variëren van klant tot klant (afhankelijk van het bedrag van de lening, interestvoet, en dergelijke) en bovendien ook nog eens over de tijd. Vaak worden dan ook heuristische methoden aangewend om de misclassificatiekosten te benaderen. Deze kunnen dan gebruikt worden om de cut-off te bepalen die toelaat de posterior kansen P(klant=goede terugbetaler | inkomen, spaarmiddelen, huwelijksstatus, ...) om te zetten naar klasse-labels.

In een eerste deel van dit doctoraat wordt dan ook uitvoerig aandacht besteed aan de grondige studie van een aantal recent voorgestelde classificatie-technieken. Hun prestatie wordt vergeleken met de klassieke technieken op een aantal real-life krediet data sets op basis van meerdere prestastiemaatstaven. Zo zullen onder meer verschillende cut-off schema's in ogenschouw genomen worden.

William van Ockham, een bekende 14de-eeuwse filosoof, benadrukte dat modellen behalve accuraat ook begrijpelijk en simpel moeten zijn. Een eenvoudig model zal immers sneller en beter in de bedrijfscontext geïntegreerd kunnen worden dan een complex, sterk geparametriseerd black-box model. Deze keuze impliceert doorgaans een trade-off omdat complexe modellen vaak ook beter presteren inzake nauwkeurigheid. Neem bijvoorbeeld neurale netwerken. Doordat deze laatste universele approximatoren zijn, leidt hun toepassing vaak tot zeer goed presterende modellen [11, 13]. Echter, een belangrijk nadeel voor de bedrijfsbesluitvorming is hun beperkte verklarende kracht: hoewel zij het mogelijk maken erg accurate uitspraken of predicties te doen, is het pijnpunt vaak dat de precieze wijze waarop zij dergelijke beslissingen afleiden niet pasklaar beschikbaar of eenvoudig interpreteerbaar is. Figuur II toont een voorbeeld van een neuraal netwerk dat getraind werd voor het schatten van de kredietwaardigheid van klanten van een financiële instelling in de Benelux: krachtig maar moeilijk interpreteerbaar. In het tweede deel van dit doctoraat wordt onder meer onderzocht hoe de neurale netwerk blackbox geopend kan worden met behulp van regelextractiemethoden. Zonder een



Figuur II: Neuraal netwerk voor het voorspellen van kredietwaardigheid.

transparanter voorstellingswijze is de kans immers groot dat de organisatie zelf onvoldoende vertrouwen heeft in de correcte werking van het model. Bovendien bestaat er in sommige landen een wettelijke verplichting inzake de openbaarheid van het gehanteerde model. Figuur III bevat de 'als-dan'-regels die uit het netwerk van Figuur II werden geëxtraheerd. Deze regels zijn eenvoudig te interpreteren en bovendien krachtig: zij blijken namelijk even accuraat als het netwerk uit Figuur II.

Naast de studie van crisp regels, zoals weergegeven in Figuur III, wordt ook het gebruik van vage (fuzzy) regels onderzocht. Figuur IV toont een voorbeeld van een verzameling vage regels voor kredietoekenning. Vage regels zijn gebruiksvriendelijker en intuïtiever dan crisp regels en kunnen dan ook interessant zijn in een kredietverleningscontext. In dit doctoraat wordt onderzocht hoe evolutionaire algoritmen en neurofuzzy systemen kunnen gebruikt worden voor het extraheren van vage regels.

Credit scoring spitst zich voornamelijk toe op het onderscheiden van goede en slechte klanten op basis van hun karakteristieken. Echter, het verschaffen van informatie betreffende het tijdstip waarop klanten wanbetaler worden kan ook heel interessant zijn voor een financiële instelling. Dit opent immers perspectieven op allerlei andere interessante toepassingen zoals het berekenen van de winst die een klant oplevert gedurende zijn lening (*profit scoring*). In dit doctoraat wordt deze problematiek, vaak omschreven als *survival analyse*, ook nader onderzocht. Hierbij worden zowel klassieke statistische methoden als neurale netwerk technieken bestudeerd.

Als Looptijd $>$ 12 maanden **En** Doel = cash provisie **En**
Spaarmiddelen $\leq$ 12.40 Euro **En** Aantal jaren klant $\leq$ 3 **Dan** Klant = slecht

Als Looptijd $>$ 12 maanden **En** Doel = cash provisie **En** Eigendom
onroerend goed = Nee **En** Spaarmiddelen $\leq$ 12.40 Euro **Dan** Klant = slecht

Als Doel = cash provisie **En** Inkomen $>$ 719 Euro **En** Eigendom
onroerend goed = Nee **En** Spaarmiddelen $\leq$ 12.40 Euro **En** Aantal jaren klant $\leq$ 3
Dan Klant = slecht

Als Doel = tweedehands wagen **En** Inkomen $>$ 719 Euro **En** Eigendom
onroerend goed = Nee **En** Spaarmiddelen $\leq$ 12.40 Euro **En** Aantal jaren klant $\leq$ 3
Dan Klant = slecht

Als Spaarmiddelen $\leq$ 12.40 Euro **En** Economische sector = Sector C
Dan Klant = slecht

Default klasse: Klant = goed

Figuur III: Als-dan'-regels geëxtraheerd uit het neuraal netwerk van Figuur II.

Als Lastenpercentage hoog is **En** Aantal jaren spaarder is klein
Dan Klant=slechte terugbetaler

Als Lastenpercentage klein is
Dan Klant=goede terugbetaler

Als Lastenpercentage medium is **En** Aantal jaren spaarder is klein
Dan Klant=slechte terugbetaler

Figuur IV: Het gebruik van vage regels voor krediettoekening.

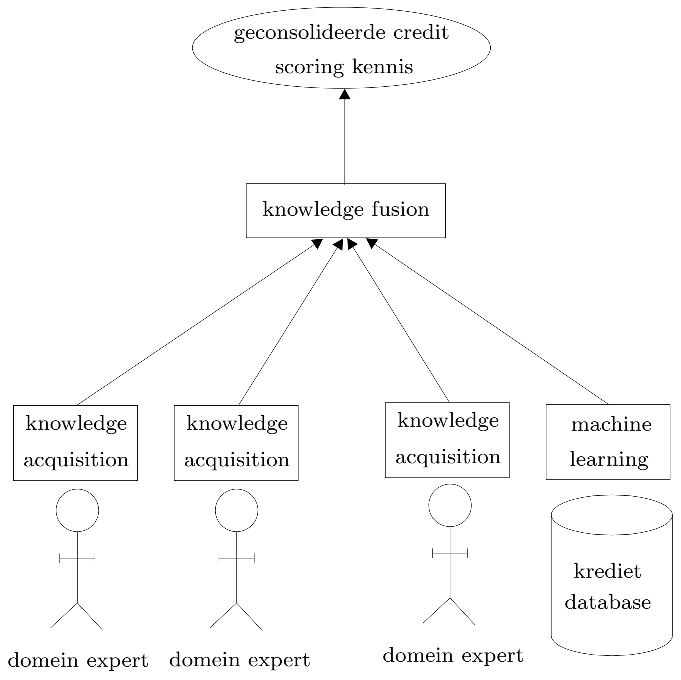
Het ontwikkelen van intelligente beslissingsondersteunende systemen voor krediettoekening

Wanneer de kennis ontgonnen is, kan de laatste stap van het KDD-proces aangevat worden. Hierbij is het de bedoeling om volledig uitrolbare beslissingsondersteunende systemen te bouwen om zo het betreffende bedrijfsproces (deels) te automatiseren. Het is hierbij belangrijk op te merken dat de visualisatie van de ontgonnen kennis heel belangrijk is om deze op een zinvolle manier te kunnen interpreteren. Verscheidene visualisatiemechanismen werden in de literatuur al voorgesteld. Hoewel OLAP (On-Line Analytical Processing) vaak in de preprocessingfase aangewend wordt, kan het ook een belangrijke rol spelen in de postprocessingfase bij de visuele weergave van, bijvoorbeeld, associatieregels [82]. In dit doctoraat wordt het gebruik van beslissingstabellen voorgesteld om de regels op een gebruiksvriendelijke manier te visualiseren [165]. Zo geeft Figuur V de beslissingstabel weer voor de regels van Figuur III. De tabel wordt typisch op een top-down manier doorlopen bij de evaluatie van nieuwe prospecten. De tabel is opgesteld met behulp van de PROLOGA software, die onder meer ook faciliteiten voorziet voor consultatie en verificatie van de geëxtraheerde regels.

1. Spaarmiddelen (Euro)	12.40													> 12.40
2. Economische sector	Sector C	andere												-
3. Doel	-	cash provisie							tweedehandswagen				ander	-
4. Looptijd	-	12 maanden				> 12 maanden			-				-	-
5. Aantal jaren klant	-	3			> 3	3	> 3		3			> 3	-	-
6. Eigendom onroerend goed	-	ja	nee		-	-	ja	nee	ja	nee		-	-	-
7. Inkomen (Euro)	-	-	719	> 719	-	-	-	-	-	719	> 719	-	-	-
1. Klant = goed	-	x	x	-	x	-	x	-	x	x	-	x	x	x
2. Klant = slecht	x	-	-	x	-	x	-	x	-	-	x	-	-	-
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Figuur V: Beslissingstabel voor de regels van Figuur III.

Een ander cruciaal aandachtspunt tijdens de laatste fase van het KDD-proces betreft de integratie van de nieuw ontgonnen kennis met de bestaande kennis. Immers, de expert - diegene die momenteel verantwoordelijk is voor de beslissingen van het bedrijfsproces - heeft ook heel wat ervaring en kennis die soms niet door het data mining algoritme gemodelleerd kunnen worden. Het is dan ook belangrijk om beide vormen van kennis, de kennis die uit data is ontgonnen en de aanwezige impliciete kennis, samen te integreren binnen één coherent en onderhoudbaar beslissingsondersteunend systeem (zie Figuur VI). Dit proces wordt in de literatuur vaak omschreven als *knowledge fusion* en vormt een uitdagend kennismanagementvraagstuk waarnaar nog heel wat praktijkgedreven onderzoek nodig is.



Figuur VI: Het *knowledge fusion* proces.

Besluit

Bij het gebruik van KDD voor de ontwikkeling van intelligente beslissingsondersteunende systemen voor kredietverlening spelen tal van aspecten een rol. In dit doctoraat benadrukken we dat modellen voor kredietoekenning idealiter zowel accuraat als begrijpelijk moeten zijn. Wat het eerste betreft, kunnen we vaststellen dat een brede waaier aan classificatie-algoritmen reeds is voorgesteld in de literatuur. Een eerste objectief van dit onderzoek beoogt dan ook de prestatie van deze algoritmen op een zinvolle en statistisch verantwoorde manier te bestuderen op een aantal real-life krediet data sets. Verder stellen we het gebruik van neurale netwerk regelextractie-technieken voor, gecombineerd met beslissingstabellen, om zo intelligente en begrijpbare beslissingsondersteunende systemen voor kredietverlening te bouwen. Naast het gebruik van crisp beslissingsregels, onderzoeken we tevens de kracht en interpreteerbaarheid van vage beslissingregels geëxtraheerd met behulp van evolutionaire algoritmen en neurofuzzy systemen. Tenslotte onderzoeken we ook het gebruik van statistische methoden en neurale netwerken voor het voorspellen van het tijdstip waarop klanten wanbetaler worden.

Chapter 1

Introduction

The last decades witnessed a spectacular increase in computer infrastructures and resources to store huge amounts of data. Many businesses have eagerly adopted these data storing facilities to record information regarding their daily operations. Examples are banks that store information regarding the repayment behavior of their customers, supermarkets that store every purchase of an individual into their data warehouses and stock exchanges that record stock prices at regular time intervals. These are all businesses where massive amounts of data are being generated and stored electronically on a daily basis.

Until recently, this data was analyzed using basic query and reporting utilities. The advent of knowledge discovery in data (KDD) technology has created the opportunity to extract more intelligent and advanced knowledge from these huge collections of data. Machine learning is one of the key technologies underlying KDD and aims at acquiring knowledge by learning patterns from data. Knowledge of these patterns could then be efficiently used to optimize sales strategies or business operations in order to gain profits or cut costs.

In this dissertation, we study the use of machine learning algorithms to develop intelligent systems for credit scoring. The problem of credit scoring is essentially a classification task which aims at distinguishing good payers from bad payers using all possible characteristics describing the applicant. It is our aim to develop decision support systems for credit scoring which are both accurate and comprehensible. Both criteria are believed to play a pivotal role in the successful deployment of automated credit scoring systems.

1.1 The Knowledge Discovery in Data Process

The quest for knowledge extracted from data has been commonly referred to as *Knowledge Discovery in Data (KDD)* and was first coined by Fayyad in 1996 as follows:

> *"...the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [84]".*

KDD is conceived as an iterative process consisting of the following steps: data preprocessing, data mining, and post-processing (see Figure 1.1). These steps are

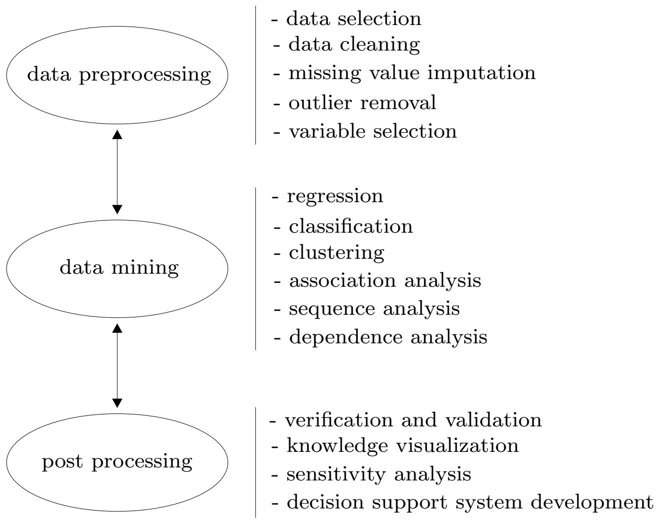


Figure 1.1: The Knowledge Discovery in Data process.

typically executed in an iterative and ad-hoc manner. In the data preprocessing phase, the data is selected and cleaned. Inconsistencies are removed and missing values and outliers are dealt with. The preprocessed data set then serves as input to the data mining phase. A distinction needs to be made between several types of data mining. This is illustrated in Table 1.1. Predictive data mining tries to predict future values of a dependent variable based on patterns learnt from past data. Regression and classification are the most popular predictive data mining tasks. These are often also described as supervised learning tasks because a dependent variable can be identified which will be used to steer the learning process. Descriptive data mining tries to identify patterns or relationships present in the data without presuming a specific dependent variable (unsupervised learning). Clustering, association analysis, sequence analysis, and dependence analysis are the most popular descriptive data mining tasks. One of the key technologies underlying

Data mining	**Task**	**Characteristic**	**Example**
predictive data mining	regression	predicting a continuous dependent variable	predicting stock prices, sales amount
	classification	predicting a categoric dependent variable	predicting credit default, bankruptcy
descriptive data mining	clustering	identifying homogeneous subpopulations	market segmentation
	association analysis	identifying relationships between items	identifying frequently bought products (market basket analysis)
	sequence analysis	identifying relationships between items over time	identifying time sequence of purchase
	dependence analysis	identifying dependencies between variables	identifying dependencies between medical parameters

Table 1.1: Types of data mining.

the data mining step is *machine learning*. Machine learning is a multidisciplinary research field providing a multitude of induction algorithms which aim at acquiring knowledge by learning patterns from data. Machine learning algorithms have been developed to tackle each of the data mining tasks depicted in Table 1.1. Some examples are given in Table 1.2. The final phase of the KDD process is

Data mining task	**Machine learning algorithm**
regression	linear regression, regression trees, neural networks, support vector machines
classification	decision trees, neural networks, k-nearest neighbor, rule induction
clustering	k-means, Kohonen neural networks
association analysis	Apriori
sequence analysis	modified Apriori
dependence analysis	Bayesian networks, graphical methods

Table 1.2: Example machine learning algorithms.

to post-process the knowledge and patterns extracted in the data mining phase. Examples of tasks that are performed here are: verification and validation of the patterns, visualization of the knowledge in alternative ways, sensitivity analysis, and integration of the extracted patterns into a decision support or knowledge application tailored to the specific business problem.

In this dissertation, we will especially focus on the data mining step and more specifically on the use of machine learning methods for classification. This will be investigated in the context of developing intelligent systems for credit scoring.

1.2 The Credit Scoring Classification Problem

Classification is one of the most frequently occurring tasks of human decision making. A classification problem arises when an object needs to be assigned to a predefined class or group according to its characteristics. A classification task is sometimes also referred to as a supervised learning task since the classes or groups are defined beforehand and can be used to steer the learning process. Many decision problems in a variety of domains such as engineering, medical science, human sciences and management science can be considered as classification problems. Popular examples are speech recognition, character recognition, medical diagnosis, bankruptcy prediction and credit scoring. Many of the ideas presented in this dissertation are equally applicable to these other domains.

One of the key decisions financial institutions have to make is to decide whether or not to grant a loan to an applicant. This basically boils down to a binary classification problem which aims at distinguishing good payers from bad payers. Until recently, this decision was made using a judgmental approach by merely inspecting the application form details of the applicant. The credit expert then decided whether the loan should be accepted or rejected using all possible relevant information describing the socio-demographic status, economic conditions and intentions of the applicant. The advent of data storage technology has facilitated financial institutions to store all information regarding the characteristics and repayment behavior of credit applicants electronically. Together with the emergence of advanced machine learning algorithms, this has motivated the need to build automated credit scoring models (also called scorecards) which aim at summarizing all available information of an applicant in a score reflecting his/her creditworthiness. If this score is above a predetermined threshold, credit is granted, otherwise credit is denied.

A multitude of machine learning algorithms have been suggested in the literature to perform credit scoring. The question then naturally arises which technique is most appropriate to build powerful scorecards. In this dissertation, we argue that two properties are essential for the successful adoption of a constructed scorecard into the daily credit decision environment. First, the scorecard should achieve a high performance in discriminating good customers from bad customers. Although this may seem an obvious criterion, it will soon become clear that measuring the accuracy of a credit scoring system is not a trivial exercise. A second important critical success factor of a scorecard is its degree of transparency and comprehensibility. In other words, a good scorecard should also be intelligent in the sense that it should provide a clear insight to the expert about how and why a particular applicant is classified as good or bad. This is one of the topics that will be thoroughly investigated in this dissertation.

Traditional credit scoring models aim at distinguishing good payers from bad payers at the time of underwriting the loan. However, the issue of when customers become bad is also very interesting to investigate since it can provide the bank with the ability to compute the profitability over a customer's lifetime. This problem statement is often referred to as survival analysis and is also one of the research issues that will be addressed in this dissertation.

1.3 Contributions

Having highlighted the need for powerful and intelligent credit scoring systems, we will now turn to the major research questions and contributions of this dissertation.

1.3.1 Benchmarking state of the art classification algorithms

In a first chapter, we start with conducting a benchmarking study comparing the classification accuracy of a number of state of the art classification algorithms on a selection of publicly available data sets. This study will include a number of recently suggested algorithms such as support vector machines and least squares support vector machines which have not yet been thoroughly compared before. A rigorous statistical setup will be employed using the appropriate test statistics to compare the performance measures. It will allow us to draw conclusions regarding the superiority of these recently suggested classifiers with respect to the other algorithms.

1.3.2 Investigating the impact of various cut-off setting schemes on scorecards

In a next chapter, the same benchmarking experiment will be repeated but hereby focussing solely on 8 real-life credit scoring data sets originating among other from major Benelux and U.K. financial institutions. In order to compute the classification accuracy of a classifier, a cut-off needs to be chosen to map a classifier's continuous output to class labels. We will experiment with various cut-off setting schemes and investigate their impact on the results. The following schemes will be considered: a cut-off of 0.5, a cut-off assuming equal sample proportions and cut-offs based on marginal good-bad rates around 5:1 and 3:1, respectively. Furthermore, we will also include the area under the receiver operating characteristic curve as a performance measure. To the best of our knowledge, such a study has not yet been conducted before in the literature.

1.3.3 Developing intelligent systems for credit scoring using neural network rule extraction and decision tables

In chapter 4, a new approach is suggested to build intelligent credit scoring systems using neural network rule extraction and decision tables. Starting from a trained and pruned neural network, we will extract rules in order to clarify its decision process. Several types of neural network rule extraction methods will be studied and compared. Alternative rule representation formalisms will be considered. In a final step, the extracted rules will be represented using decision tables. A fully deployable decision support system will then be built using the Prologa decision table workbench. The methodology to develop decision support systems for credit scoring using neural network rule extraction combined with decision tables is a major contribution of this dissertation.

1.3.4 Developing intelligent systems for credit scoring using fuzzy rule extraction

Chapter 5 studies the use of fuzzy rules for credit scoring. Fuzzy rules are believed to be more comprehensible than their crisp counterparts because the rules are expressed in linguistic, vague terms which are more close to human thinking and reasoning. Both evolutionary and neurofuzzy algorithms are investigated for fuzzy rule extraction on a number of data sets. The innovative contributions of this chapter are, the characterization of the difference between approximate and descriptive fuzzy rules, the comparative study between the classification performance of evolutionary and neurofuzzy systems for fuzzy rule extraction, and the investigation of the suitability of the extracted fuzzy rules to build intelligent, user-friendly credit scoring systems.

1.3.5 Using a neural network based approach for predicting customer default times

In a final chapter, we discuss the use of survival analysis methods for predicting when customers default. We start with reviewing the well-known statistical methods for doing survival analysis. We then indicate their shortcomings and present the use of neural networks as an interesting alternative. After an extensive literature review, we present a new type of neural network for doing survival analysis for credit scoring. The network is trained using the automatic relevance determination (ARD) extension of the Bayesian evidence framework of MacKay. This will allow us to infer the importance of the inputs in a straightforward way. This is also one of the major contributions of this dissertation.

1.4 Notation

A scalar $x \in \mathrm{I\!R}$ is denoted in normal script. Vectors are represented in boldface notation and are assumed to be column vectors: $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$. The corresponding row vector is obtained using the transpose T: $\mathbf{x^T} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}^T = [x_1 \; x_2 \; \dots \; x_n]$. Bold capital notation is used for matrices: $\mathbf{X}$. We use n for the number of inputs and N for the number of observations in a data set. The observation i is denoted as $\mathbf{x}_i$ whereas variable j is indicated as x_j. The value of variable j for observation i is represented as $\mathbf{x}_i(j)$. We use P to denote a probability and p to denote a probability density. We use the notation $|\mathbf{M}|$ to represent the determinant of matrix $\mathbf{M}$.

Chapter 2

An Overview of Classification Techniques and Issues

In this chapter, we start with explaining the basic concepts and functioning of a selection of well-known classification methods[1,2,3]*. We hereby discuss statistical classifiers (logistic regression, linear, Fisher and quadratic discriminant analysis), linear programming, Bayesian networks, decision trees and rules(C4.5), k-nearest neighbor classifiers, neural networks and (least squares) support vector machines. Although still other classification methods have been presented in the literature, we believe we discuss the most important ones, each originating from their own specific background.*

Once the classification techniques have been thoroughly discussed, we further elaborate on some issues and problems which also need to be addressed when developing classification models. We start with discussing several ways of splitting up the data in order to assess the predictive performance of a classifier. Next, we discuss the problem of input selection which aims at reducing the number of inputs of a classifier without degrading its predictive performance. In a following

[1]B. Baesens, S. Viaene, D. Van den Poel, J. Vanthienen, G. Dedene, Using bayesian neural networks for repeat purchase modelling in direct marketing, *European Journal of Operational Research*, 138(1), pp. 191-211, 2002.

[2]B. Baesens, G. Verstraeten, D. Van den Poel, M. Egmont-Petersen, P. Van Kenhove, J. Vanthienen, Bayesian network classifiers for identifying the slope of the customer lifecycle of long-life customers, *European Journal of Operational Research*, forthcoming, 2003.

[3]T. Van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, J. Vandewalle, Benchmarking Least Squares Support Vector Machine Classifiers, *Machine Learning*, forthcoming, 2003.

section, we present a first benchmarking study which compares the classification accuracy of the discussed classification techniques on 10 publicly available real-life data sets. A second benchmarking study investigates the predictive power of some of the discussed classification techniques on two artificially generated, highly non-linear classification problems. We then thoroughly discuss the shortcomings of both benchmarking studies and identify the need for the area under the receiver operating characteristic curve (AUC) as an additional performance measure. Test statistics to compare the classification accuracy and the AUC are also discussed.

2.1 Classification Techniques

2.1.1 Logistic Regression

Given a training set of N data points $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, with input data $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{0,1\}$, the logistic regression approach to classification (LOG) tries to estimate the probability $P(y=1|\mathbf{x})$ as follows:

$$P(y=1|\mathbf{x}) = \frac{1}{1+\exp(-(w_0+\mathbf{w}^T\mathbf{x}))}, \tag{2.1}$$

and,

$$P(y=0|\mathbf{x}) = \frac{\exp(-(w_0+\mathbf{w}^T\mathbf{x}))}{1+\exp(-(w_0+\mathbf{w}^T\mathbf{x}))}, \tag{2.2}$$

whereby $\mathbf{x} \in \mathbb{R}^n$ is a n-dimensional input vector, $\mathbf{w}$ is the parameter vector and the scalar w_0 the intercept.

One typically employs a maximum likelihood procedure to estimate the parameters w_0 and $\mathbf{w}$ of the logistic regression classifier. The probability of observing either class is given by

$$p(y|\mathbf{x}) = P(y=1|\mathbf{x})^y(1-P(y=1|\mathbf{x}))^{1-y}, \tag{2.3}$$

which is essentially a Bernouilli distribution [26]. The likelihood of observing the data set D, given the observations are drawn independently from 2.3 is then given by:

$$\prod_{i=1}^{N} P(y_i=1|\mathbf{x}_i)^{y_i}(1-P(y_i=1|\mathbf{x}_i))^{1-y_i}. \tag{2.4}$$

The log-likelihood function then becomes

$$LL = \sum_{i=1}^{N} y_i \log(P(y_i=1|\mathbf{x}_i)) + (1-y_i)\log(1-P(y_i=1|\mathbf{x}_i)). \tag{2.5}$$

This log-likelihood function can then be maximized using the Newton-Raphson algorithm [112]. Note that maximizing the log-likelihood LL is the same as minimizing the negative log-likelihood $-LL$. The latter is often referred to as the cross-entropy error function [26].

In order to obtain a linear decision boundary, it is required that either $P(y = 1|\mathbf{x})$ or a monotone transformation of $P(y = 1|\mathbf{x})$ is linear in $\mathbf{x}$ [112]. Since

$$\log(\frac{P(y = 1|\mathbf{x})}{1 - P(y = 1|\mathbf{x})}) = w_0 + \mathbf{w}^T\mathbf{x}, \tag{2.6}$$

the logistic regression classifier assumes a linear decision boundary modeled by the hyperplane $\{\mathbf{x}|w_0 + \mathbf{w}^T\mathbf{x} = 0\}$. Note that the term $\frac{P(y=1|\mathbf{x})}{1-P(y=1|\mathbf{x})}$ is often referred to as the odds in favor of $y = 1$.

One of the advantages of the logistic regression classifier is that it is a non-parametric technique because no assumptions are made concerning the probability distribution of the attributes.

2.1.2 Discriminant Analysis

Discriminant analysis assigns an observation $\mathbf{x}$ to the class $i \in \{0, 1\}$ having the largest posterior probability $P(y = i|\mathbf{x})$. It hereby uses Bayes'theorem to compute the posterior probability [26, 67, 112, 240]:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}. \tag{2.7}$$

The Bayesian classification rule (Bayes'rule) then says: decide $y = 1$ if one of the following conditions is met:

$$P(y = 1|\mathbf{x}) > P(y = 0|\mathbf{x}), \tag{2.8}$$

or

$$p(\mathbf{x}|y = 1)P(y = 1) > p(\mathbf{x}|y = 0)P(y = 0), \tag{2.9}$$

or

$$\log p(\mathbf{x}|y = 1) - \log p(\mathbf{x}|y = 0) > \log P(y = 0) - \log P(y = 1). \tag{2.10}$$

When one assumes the class-conditional distributions $p(\mathbf{x}|y = i), i \in \{0, 1\}$, are multivariate Gaussian

$$p(\mathbf{x}|y = i) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}_i|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T\mathbf{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)\}, \tag{2.11}$$

with $\boldsymbol{\mu}_i$ the mean vector of class i and $\boldsymbol{\Sigma}_i$ the covariance matrix of class i, the classification rule 2.10 becomes: decide $y = 1$ if:

$$\begin{aligned}&(\mathbf{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)-(\mathbf{x}-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_0^{-1}(\mathbf{x}-\boldsymbol{\mu}_0)<2(\log(P(y=1))-\log(P(y=0)))\\&+\log|\boldsymbol{\Sigma}_0|-\log|\boldsymbol{\Sigma}_1|,\end{aligned} \tag{2.12}$$

The presence of the quadratic terms $\mathbf{x}^T\boldsymbol{\Sigma}_1^{-1}\mathbf{x}$ and $-\mathbf{x}^T\boldsymbol{\Sigma}_0^{-1}\mathbf{x}$ indicates that the decision boundary is quadratic in $\mathbf{x}$ and therefore this classification technique is called *quadratic discriminant analysis* (QDA). Note that the left hand side of Equation 2.12 measures the difference in Mahalanobis distance between $\mathbf{x}$ and $\boldsymbol{\mu}_1$ and $\mathbf{x}$ and $\boldsymbol{\mu}_0$ but needs to be adapted according to the class proportions and the covariance matrices in order to make a classification decision.

A simplification occurs if $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$. In this case, the quadratic terms $\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}$ and $-\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}$ cancel and after rearranging Equation 2.12, the classification rule becomes: decide $y = 1$ if:

$$(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})>\log(P(y=0))-\log(P(y=1)), \tag{2.13}$$

with $\boldsymbol{\mu} = \frac{\boldsymbol{\mu}_0+\boldsymbol{\mu}_1}{2}$. Equation 2.13 is linear in $\mathbf{x}$ and the corresponding classification technique is called *linear discriminant analysis* (LDA).

Since we usually do not know the exact values of $\boldsymbol{\mu}_0$, $\boldsymbol{\mu}_1$, $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Sigma}_1$, they have to be estimated from the data sample $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$. For LDA, the estimates for $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Sigma}_1$ are typically pooled into one estimate for the common covariance matrix $\boldsymbol{\Sigma}$. Many statistical tests have been suggested to test the equality of the covariance matrices and choose between LDA and QDA. The Bartlett test [217] and Levene test [147] are amongst the most popular.

Fisher discriminant analysis is another popular classification technique which aims at finding a linear combination of the variables that exhibits the largest difference in the projected group means $\mathbf{w}^T\boldsymbol{\mu}_1$ and $\mathbf{w}^T\boldsymbol{\mu}_0$ relative to the projected within class variance $\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}$ by maximizing the following expression[87]:

$$\text{argmax}_{\mathbf{w}}\frac{(\mathbf{w}^T\boldsymbol{\mu}_1-\mathbf{w}^T\boldsymbol{\mu}_0)^2}{\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}}, \tag{2.14}$$

or

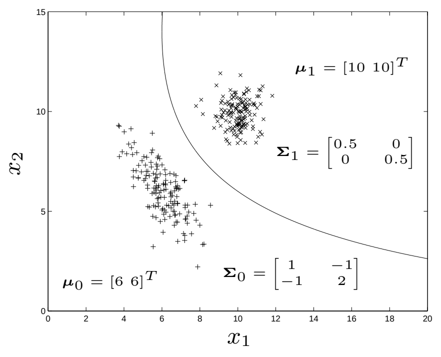
$$\begin{aligned}&\text{argmax}_{\mathbf{w}}(\mathbf{w}^T\boldsymbol{\mu}_1-\mathbf{w}^T\boldsymbol{\mu}_0)^2\\&\text{subject to}:\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}=1.\end{aligned} \tag{2.15}$$

Mathematical derivation then yields $\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$. The classification rule then becomes: decide $y = 1$ if:

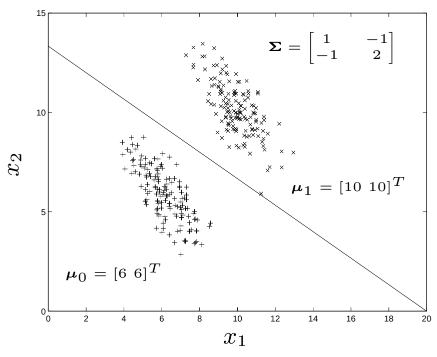
$$|\mathbf{w}^T\mathbf{x}-\mathbf{w}^T\boldsymbol{\mu}_1|<|\mathbf{w}^T\mathbf{x}-\mathbf{w}^T\boldsymbol{\mu}_0|, \tag{2.16}$$

or,

$$(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})>0, \tag{2.17}$$



(a) QDA



(b) LDA

Figure 2.1: Quadratic versus Linear Discriminant analysis.

which is the same as 2.13 when $P(y = 0) = P(y = 1)$.

Both LDA and QDA are popular classification techniques which have been successfully applied in various settings. In most cases reported in the literature LDA outperforms QDA. One major reason for this is that in the QDA case more parameters need to be estimated from the same sample and these estimates may be poor for small data sets with many inputs [92].

The choice between discriminant analysis and logistic regression strongly depends upon the characteristics of the data. If a data set contains many qualitative inputs, the multivariate normality assumption no longer holds and a logistic regression classifier will probably give the best performance [69, 183]. On the other hand, when the inputs are quantitative and multivariate normally distributed, discriminant analysis may proof to be the best classification technique.

Example 2.1
Consider a binary classification case with $\boldsymbol{\mu}_0 = [6\ 6]^T$, $\boldsymbol{\mu}_1 = [10\ 10]^T$, $\boldsymbol{\Sigma}_0 = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$, $\boldsymbol{\Sigma}_1 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ and with equal prior class probabilities ($P(y = 1) = P(y = 0) = 0.5$). Since the covariance matrices are different, the decision boundary will be quadratic as illustrated in the left pane of Figure 2.1. However, when assuming one common covariance matrix $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$, a linear decision boundary is obtained (see right pane of Figure 2.1).

2.1.3 Linear Programming

Linear programming (LP) is probably one of the most commonly used techniques for credit scoring in the industry nowadays. Since the pioneering work of Mangasarian [158], numerous LP methods for classification have been suggested in the literature. A very popular formulation goes as follows [234, 235]:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \sum_{i=1}^{N} \xi_i \tag{2.18}$$

subject to

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i \geq c - \xi_i, & y_i = +1 \\ \mathbf{w}^T\mathbf{x}_i \leq c + \xi_i, & y_i = -1 \\ \xi_i \geq 0, & i = 1, ..., N, \end{cases} \tag{2.19}$$

whereby $\boldsymbol{\xi}$ represents the vector of ξ_i values. The first (second) set of inequalities tries to separate the goods (bads) from the bads (goods) by assigning them a score $\mathbf{w}^T\mathbf{x}_i$ which is higher (lower) than the prespecified cut-off c. However, since one has to take into account misclassifications, the positive slack variables ξ_i are entered. The aim is then to minimize the misclassifications by minimizing the sum of the slack variables ξ_i. Note that many variants of this method have been proposed which use other cut-off strategies, objective functions, and/or require some variables to be integer [90, 96, 235].

One of the advantages of using LP methods for credit scoring is that they can easily model domain knowledge or a priori bias by including additional constraints [235]. If, e.g., one knows a priori that variable x_k has a larger impact on the score than variable x_l, one can add the constraint $w_k \geq w_l$ to the program. A major difficulty with LP methods is the lack of a solid statistical underpinning which makes it rather difficult to decide upon the statistical significance of the estimated parameters and perform input selection. In [76], it was empirically shown that LP methods do not perform quite as well as statistical regression methods.

2.1.4 Bayesian Networks for Classification

A Bayesian network (BN) represents a joint probability distribution over a set of discrete, stochastic variables. It is to be considered as a probabilistic white-box model consisting of a qualitative part specifying the conditional (in)dependencies between the variables and a quantitative part specifying the conditional probabilities of the data set variables [176]. Formally, a Bayesian network consists of two parts $B = \langle G, \Theta \rangle$. The first part G is a directed acyclic graph consisting of nodes and arcs. The nodes are the variables $x_1, ..., x_n$ in the data set whereas the arcs indicate direct dependencies between the variables. The graph G then encodes the independence relationships in the domain under investigation. The second part of the network, Θ, represents the conditional probability distributions. It contains a parameter $\theta_{x_j|\Pi(x_j)} = P_B(x_j|\Pi(x_j))$ for each possible value of x_j, given each combination of the set of direct parent variables of x_j in G, $\Pi(x_j)$. The network B then represents the following joint probability distribution:

$$P_B(x_1, ..., x_n) = \prod_{j=1}^{n} P_B(x_j|\Pi(x_j)) = \prod_{j=1}^{n} \theta_{x_j|\Pi(x_j)}. \tag{2.20}$$

The first task when learning a Bayesian network is to find the structure G of the network. Once we know the network structure G, the parameters Θ need to be estimated. In general, these two estimation tasks are performed separately. One commonly uses the empirical frequencies from the data D to estimate these parameters [4]:

$$\theta_{x_j|\Pi(x_j)} = \hat{P}_D(x_j|\Pi(x_j)). \tag{2.21}$$

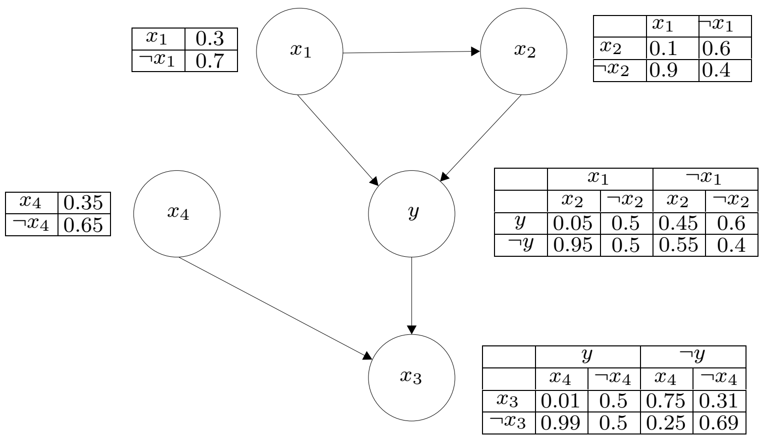
It can be shown that these estimates maximize the log likelihood of the network B given the data D [94]. Note that these estimates might be further improved by a smoothing operation [94].

A Bayesian network is essentially a statistical model that makes it feasible to compute the (joint) posterior probability distribution of any subset of unobserved stochastic variables, given that the variables in the complementary subset are observed. This functionality makes it possible to use a Bayesian network as a statistical classifier by applying the winner-takes-all rule to the posterior probability distribution for the (unobserved) class node [66]. The underlying assumption behind the winner-takes-all rule is that all gains and losses are equal (for a discussion of this aspect see, e.g., [66]). Consider the Bayesian network depicted in Figure 2.2 (taken from [73]). It consists of four variables x_1, x_2, x_3, x_4 and one classification node y. The variables are all binary and can be either true or false. The conditional probability tables are also given. Suppose we want to compute the probability $P(y \mid x_1, x_2, \neg x_3, x_4)$. From the definition of conditional probability, we have:

$$\begin{aligned} P(y \mid x_1, x_2, \neg x_3, x_4) &= \tfrac{P(y,x_1,x_2,\neg x_3,x_4)}{P(x_1,x_2,\neg x_3,x_4)} \\ &= \tfrac{P(y,x_1,x_2,\neg x_3,x_4)}{P(y,x_1,x_2,\neg x_3,x_4)+P(\neg y,x_1,x_2,\neg x_3,x_4)}. \end{aligned} \tag{2.22}$$

By looking at the graph and using equation 2.20, we have $P(y, x_1, x_2, \neg x_3, x_4) = P(x_1)P(x_2 \mid x_1)P(y \mid x_1, x_2)P(x_4)P(\neg x_3 \mid y, x_4)$, or, by using the probability tables, $P(y, x_1, x_2, \neg x_3, x_4) = 0.3 \times 0.1 \times 0.05 \times 0.35 \times 0.99 = 0.00051975$. Likewise, we can find that $P(\neg y, x_1, x_2, \neg x_3, x_4) = 0.3 \times 0.1 \times 0.95 \times 0.35 \times 0.25 = 0.00249375$. Hence, from equation 2.22, we have $P(y \mid x_1, x_2, \neg x_3, x_4) = \frac{0.00051975}{0.00051975+00249375} = 0.1725$, and $P(\neg y \mid x_1, x_2, \neg x_3, x_4) = 0.8275$.

[4]Note that we hereby assume that the data set is complete, i.e., no missing values.



x_1	0.3
$\neg x_1$	0.7

	x_1	$\neg x_1$
x_2	0.1	0.6
$\neg x_2$	0.9	0.4

x_4	0.35
$\neg x_4$	0.65

	x_1		$\neg x_1$	
	x_2	$\neg x_2$	x_2	$\neg x_2$
y	0.05	0.5	0.45	0.6
$\neg y$	0.95	0.5	0.55	0.4

	y		$\neg y$	
	x_4	$\neg x_4$	x_4	$\neg x_4$
x_3	0.01	0.5	0.75	0.31
$\neg x_3$	0.99	0.5	0.25	0.69

Figure 2.2: An example Bayesian network classifier [73].

When one or more variables are missing, the posterior probability of each unobserved node can be computed using advanced algorithms (e.g., [145]). However, this is less likely to occur in a classification setting.

Example 2.2
In [14, 253], several types of Bayesian network classifiers were adopted to predict the sign of the customer lifecycle slope which indicates whether the customer will increase or decrease his/her future spending from initial purchase information. Figure 2.3 depicts a Bayesian network classifier that was learned from UPC scanner data obtained from a large Belgian Do-It-Yourself retail chain. The total contribution of the client (TotCont),

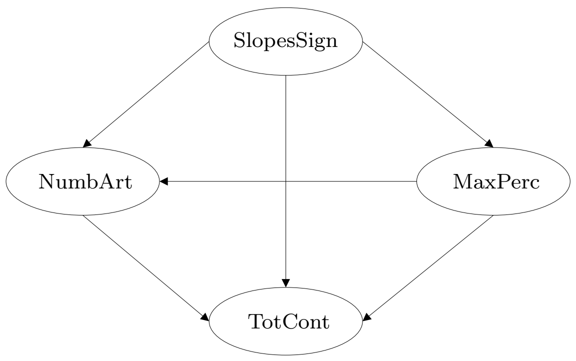


Figure 2.3: Example of a Bayesian network classifier for marketing [14].

the total number of articles bought (NumbArt) and the maximum percentage of products bought in one product family (MaxPerc) proved to be very powerful predictors for predicting the sign of the customer lifecycle slope.

Example 2.3
In [8], the power and usefulness of Bayesian network classifiers for credit scoring was evaluated. Various types of Bayesian network classifiers were evaluated and contrasted including unrestricted Bayesian network classifiers learned using Markov Chain Monte Carlo (MCMC) search. Figure 2.4 provides an example of an (unrestricted) Bayesian network that was learned for the publicly available German credit data set (see chapter 3). It was shown that MCMC Bayesian network classifiers yielded a very good performance and by using the Markov blanket concept[5], a natural form of feature selection was obtained, which resulted in parsimonious and powerful models for financial credit scoring.

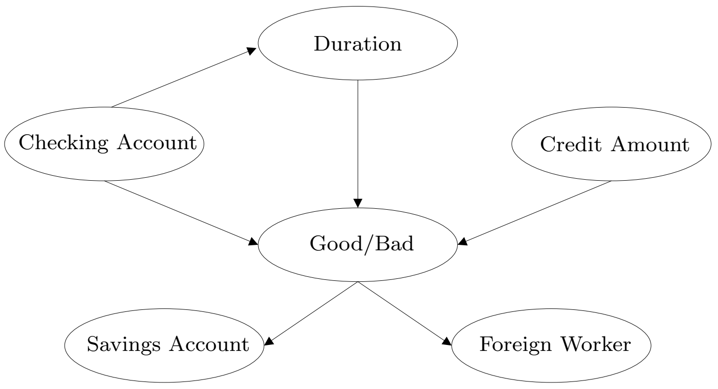


Figure 2.4: Example of a Bayesian network classifier for credit scoring [8].

2.1.5 The naive Bayes classifier

A simple classifier, which in practice often performs surprisingly well, is the naive Bayes classifier [66, 130, 143]. This classifier basically learns the class-conditional probabilities $p(x_j|y)$ of each variable x_j given the class label y. A new test case $\mathbf{x}$ is then classified by using Bayes' rule to compute the posterior probability of each class y given the vector of observed variable values:

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})}. \tag{2.23}$$

The simplifying assumption behind the naive Bayes classifier is that the variables are conditionally independent given the class label. Hence,

$$p(\mathbf{x}|y) = \prod_{j=1}^{n} p(x_j|y). \tag{2.24}$$

[5]If the direct parents of the classification node C are denoted by Π_C and the direct children by Σ_C, the Markov blanket of node C is given by $\Pi_C \cup \Sigma_C \cup \Pi_{\Sigma_C}$. It can be shown that once all variables in the Markov blanket of the classification node C are observed, it is independent from any other remaining variables in the network.

This assumption simplifies the estimation of the class-conditional probabilities from the training data. Notice that one does not estimate the denominator in expression 2.23 since it is independent of the class. Instead, one normalizes the numerator term $p(y)p(\mathbf{x}|y)$ to 1 over all classes. Naive Bayes classifiers are easy to construct since the structure is given a priori and no structure learning phase is required. The probabilities $p(x_j|y)$ are estimated by using the frequency counts for the discrete variables and a normal or kernel density based method for continuous variables [130]. Figure 2.5 provides a graphical representation of a naive Bayes classifier.

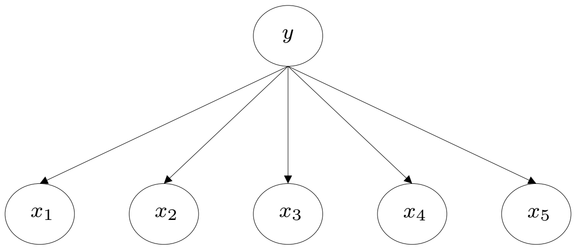


Figure 2.5: The naive Bayes classifier.

2.1.6 Tree augmented naive Bayes classifiers

In [94], tree augmented naive Bayes classifiers (TANs) were presented as an extension of the naive Bayes Classifier. TANs relax the independence assumption by allowing arcs between the variables. An arc from variable x_k to x_l then implies that the impact of x_k on the class variable also depends on the value of x_l. An example of a TAN is presented in Figure 2.6. In a TAN network the class variable has no parents and each variable has as parents the class variable and at most one other variable. The variables are thus only allowed to form a tree structure. In [94], a procedure was presented to learn the optional arrows in the structure that forms a TAN network. This procedure is based on an earlier algorithm suggested by Chow and Liu (CL) [46]. The procedure consists of the following five steps.

1. Compute the conditional mutual information given the class variable y, $I(x_k; x_l|y)$, between each pair of variables, $k \neq l$. $I(x_k; x_l|y)$ is defined as follows:
$$I(x_k; x_l|y) = \sum p(x_k, x_l, y)\log\frac{p(x_k, x_l|y)}{p(x_k|y)p(x_l|y)}. \qquad (2.25)$$
This function is an approximation of the information that x_l provides about x_k (and vice versa) when the value of y is known.

2. Build a complete undirected graph in which the nodes are the variables. Assign to each arc connecting x_k to x_l the weight $I(x_k; x_l|y)$.

3. Build a maximum weighted spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all arcs to be outward from it.
5. Add the classification node y and draw an arc from y to each x_k.

We will use Kruskal's algorithm in step 3 to construct the maximum weighted spanning tree [141]. In [94], it was proven that the above procedure builds TANs that maximize the log likelihood of the network given the training data and has time complexity $O(n^2 \cdot N)$ with n the number of variables and N the number of data points. Experimental results indicated that TANs outperform naive Bayes with the same computational complexity and robustness [94].

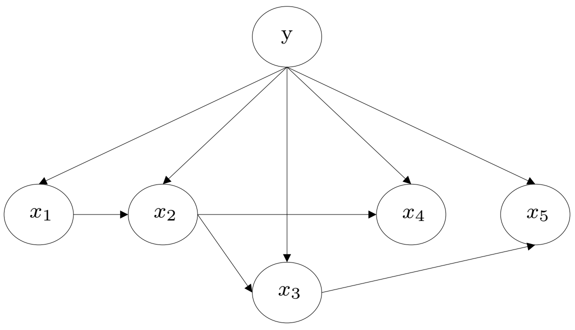


Figure 2.6: The tree augmented naive Bayes classifier.

2.1.7 Decision Trees and Rules

A decision tree represents a structure with two types of components [187]:

- leaf nodes that assign class labels to observations;
- internal nodes that specify tests on individual attributes with one branch and subtree for each outcome of the test.

The tree classifies observations in a top-down manner, starting from the root and working one's way down according to the outcomes of the tests at the internal nodes, until a leaf node has been reached and a class label has been assigned. Figure 2.7 presents an example of a decision tree taken from [163]. The tree classifies Saturday mornings as suitable or not for playing tennis.

Many decision tree and rule induction algorithms have already been suggested in the literature [33]. One of the most popular is the C4.5 algorithm [187]. C4.5

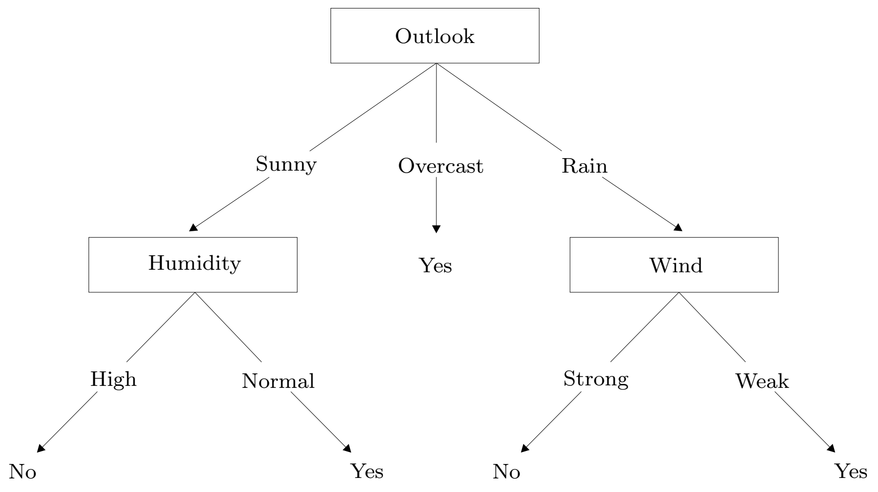


Figure 2.7: Example Decision Tree [163].

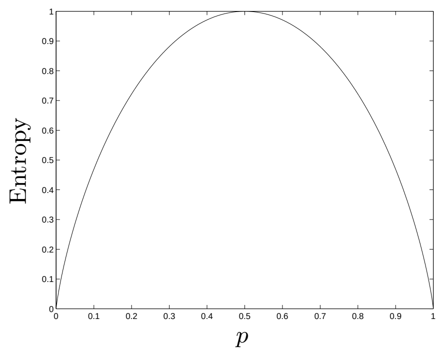


Figure 2.8: The Entropy measure.

induces decision trees based on information theoretic concepts. Let p_1 (p_0) be the proportion of examples of class 1 (0) in sample S. The entropy of S is then calculated as follows:

$$\text{Entropy}(S) = -p_1 \log_2(p_1) - p_0 \log_2(p_0), \tag{2.26}$$

whereby $p_0 = 1 - p_1$. Entropy is used to measure how informative an attribute is in splitting the data. Figure 2.8 provides a graphical representation of the Entropy measure. Basically, the entropy measures the order (or disorder) in the data with respect to the classes. It equals 1 when $p_1 = p_0 = 0.5$ (maximal disorder, minimal order) and 0 (maximal order, minimal disorder) when $p_1 = 0$ or $p_0 = 0$. In the latter case, all observations belong to the same class.

$\text{Gain}(S, x_j)$ is defined as the expected reduction in entropy due to sorting (splitting) on attribute x_j:

$$\text{Gain}(S, x_j) = \text{Entropy}(S) - \sum_{v\,\in\,\text{values}(x_j)} \frac{|S_v|}{|S|}\text{Entropy}(S_v), \tag{2.27}$$

where $\text{values}(x_j)$ represents the set of all possible values of attribute x_j, S_v the subset of S where attribute x_j has value v and $|S_v|$ the number of observations in S_v. The Gain criterion was used in ID3, the forerunner of C4.5, to decide upon which attribute to split at a given node [186]. However, when this criterion is used to decide upon the node splits, the algorithm favors splits on attributes with many distinct values. Hence, when the data set contains an ID attribute with a distinct value for each observation, the Gain criterion will select it as the best splitting decision. In order to rectify this, C4.5 applies a normalization and uses the gainratio criterion which is defined as follows:

$$\begin{aligned} \text{Gainratio}(S, x_j) &= \frac{\text{Gain}(S, x_j)}{\text{SplitInformation}(S, x_j)} \text{ with} \\ \text{SplitInformation}(S, x_j) &= -\sum_{k\,\in\,\text{values(x}_\text{j})} \frac{|S_k|}{|S|}\log_2\frac{|S_k|}{|S|}. \end{aligned} \tag{2.28}$$

The SplitInformation is the entropy of S with respect to the values of x_j. It measures how broadly and uniformly attribute x_j splits the data and it discourages the selection of attributes with many uniformly distributed values [163]. Consider e.g. the ID example whereby each instance has a distinct ID. If there are N instances, the SplitInformation will be $\log_2(N)$. However, if we consider a boolean attribute which splits the N instances exactly in half, the SplitInformation will be 1. If both attributes have the same Gain, the latter will obviously have a higher Gainratio and will be selected [163]. However, when $|S_k| \approx |S|$ for an attribute which has the same value for most of the observations in S, the SplitInformation will be very small and thus the Gainratio may become very large. To counter this effect, C4.5 first computes the Gain of all attributes and then selects the attribute with the highest Gainratio, subject to the constraint that its Gain is at least as large as the average Gain over all attributes examined.

The tree is then constructed by means of recursive partitioning until the current leaf nodes contain only instances of a single class or until no test offers any improvement. However, since most real-life data sets are noisy, and since in most cases the attributes have limited predictive power, this tree growing strategy often results in a complex tree with many internal nodes that overfits the data. C4.5 tries to remedy this phenomenon by a pruning procedure that is executed retrospectively once the full tree has been grown. It hereby uses a heuristic based on the binomial distribution (see [187] for more details)

The unpruned C4.5 tree can be easily converted into a rule set by deriving a rule for each path from the root of the unpruned tree to a leaf node (C4.5rules).

These rules may then be further pruned by removing conditions based on a similar procedure as with the tree. Since the rules are no longer mutually exclusive and exhaustive after pruning, C4.5 orders them heuristically and also chooses a default class.

2.1.8 K-Nearest Neighbour Classifiers

K-nearest neighbour classifiers (KNN) classify a data instance by considering only the k most similar data instances in the training set [1, 67, 112]. The class label is then assigned according to the class of the majority of the k-nearest neighbours. The choice of k can be made using a cross-validation method. Ties can be avoided by choosing k odd or assigning the observation to the default class. Note that it is important to first normalize the inputs to e.g. mean 0 and standard deviation 1 since they may have been measured in different units. One commonly opts for the Euclidean distance as the similarity measure:

$$d(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{x}_i - \mathbf{x}_j|| = [(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)]^{1/2}, \tag{2.29}$$

where $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^n$ are the input vectors of data instance i and j, respectively. Henley and Hand [117] propose the use of an adjusted Euclidean metric:

$$d_a(\mathbf{x}_i, \mathbf{x}_j) = [(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{I} + d\mathbf{w}\mathbf{w}^T)(\mathbf{x}_i - \mathbf{x}_j)]^{1/2}, \tag{2.30}$$

where $\mathbf{I}$ represents the identity matrix, $\mathbf{w}$ the weights obtained from Fisher discriminant analysis or logistic regression, and d a parameter which needs to be tuned individually for each data set. For an overview of other metrics used in k-nearest neighbor classification, see e.g. [67, 112].

Example 2.4
Figure 2.9 illustrates the 5-nearest neighbor classification rule. Starting from the test point $\mathbf{x}$, the classifier grows a spherical region until the 5 nearest training points are enclosed. The classifier will then assign the label $\times$ to $\mathbf{x}$ since 3 of the 5 nearest neighbors have this label [67].

It is possible to compute theoretical asymptotic upperbounds on the performance of the k-nearest neighbor classifier [67, 112]. A disadvantage of the k-nearest neighbor classifier is its large computing power requirement, since for classifying an object its distance to all the objects in the training set has to be calculated. Furthermore, when many irrelevant attributes are present, the classification performance may degrade when observations have distant values for these attributes.

2.1.9 Neural Networks

Neural networks (NNs) are mathematical representations inspired by the functioning of the human brain. Many types of neural networks have been suggested in

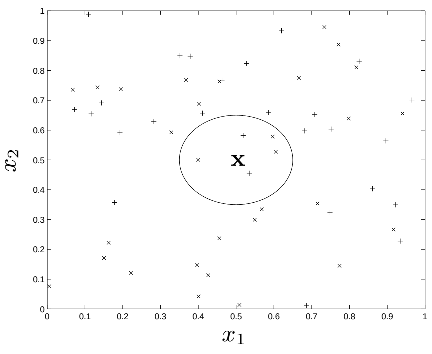


Figure 2.9: The 5-nearest neighbor classifier [67].

the literature for both supervised and unsupervised learning [26, 193, 267]. Since our focus is on classification, we will discuss the Multilayer Perceptron (MLP) neural network in more detail because it is the most popular neural network for classification.

A MLP is typically composed of an input layer, one or more hidden layers and an output layer, each consisting of several neurons. Each neuron processes its inputs and generates one output value which is transmitted to the neurons in the subsequent layer. One of the key characteristics of MLPs is that all neurons and layers are arranged in a feedforward manner and no feedback connections are allowed. Figure 2.10 provides an example of an MLP with one hidden layer and one output neuron.

The output of hidden neuron i is then computed by processing the weighted inputs and its bias term $b_i^{(1)}$ as follows:

$$h_i = f^{(1)}(b_i^{(1)} + \sum_{j=1}^{n} \mathbf{W}_{ij} x_j). \tag{2.31}$$

$\mathbf{W}$ is the weight matrix whereby $\mathbf{W}_{ij}$ denotes the weight connecting input j to hidden unit i. In an analogous way, the output of the output layer is computed as follows:

$$z = f^{(2)}(b^{(2)} + \sum_{j=1}^{n_h} \mathbf{v}_j h_j), \tag{2.32}$$

with n_h the number of hidden neurons and $\mathbf{v}$ the weight vector whereby $\mathbf{v}_j$ represents the weight connecting hidden unit j to the output neuron. The bias inputs play a similar role as the intercept term in a classical linear regression model. A threshold function is then typically applied to map the network output y to a classification label. The transfer functions $f^{(1)}$ and $f^{(2)}$ allow the network to model non-linear relationships in the data. Examples of transfer functions that

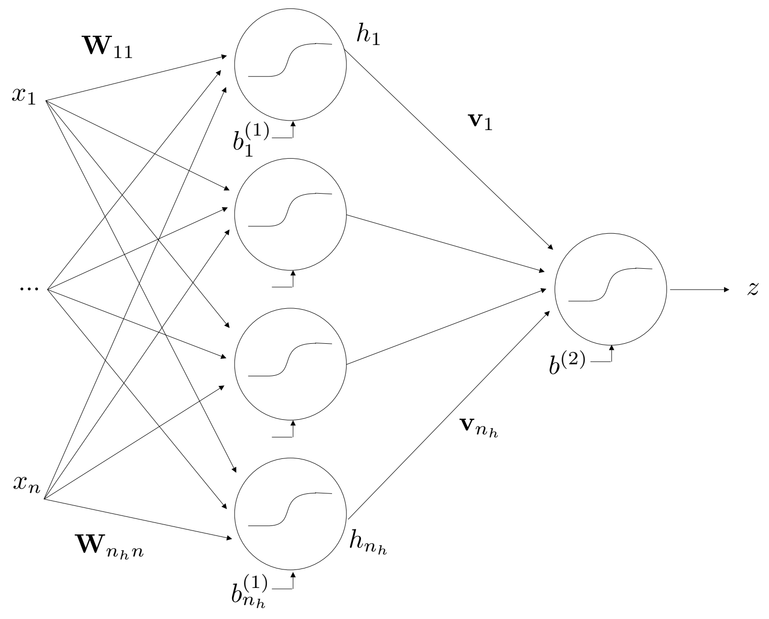
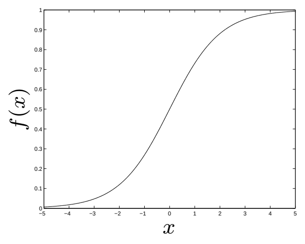


Figure 2.10: Architecture of a Multilayer Perceptron with one hidden layer.

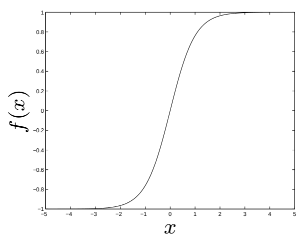
are commonly used are the sigmoid $f(x) = \frac{1}{1+\exp(-x)}$, the hyperbolic tangent $f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$ and the linear transfer function $f(x) = x$ (see Figure 2.11). For a binary classification problem, it is convenient to use the logistic transfer function in the output layer ($f^{(2)}$), since its output is limited to a value within the range $[0, 1]$. This allows the output y of the MLP to be interpreted as a conditional probability of the form $P(y = 1|\mathbf{x})$[26, 192]. In that way, the neural network naturally produces a score per data instance, which allows the data instances to be ranked accordingly for scoring purposes (e.g. customer scoring).

Note that multiple hidden layers might be used but theoretical works have shown that NNs with one hidden layer are universal approximators capable of approximating any continuous function to any desired degree of accuracy on a compact interval (*universal approximation property*) [26, 123]. Remark that one might also use M output neurons for an M-class classification problem whereby the class is then assigned according to the output neuron with the highest output value (*winner take all learning*) [26].

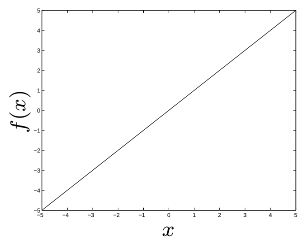
The weights $\mathbf{W}$ and $\mathbf{v}$ are the crucial parameters of the network and need to be estimated during a learning process. Given a training set of N data points D= $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, with input data $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{0, 1\}$, the weights of the network are first randomly initialized and then iteratively adjusted so as to minimize an objective function, typically the sum of



(a) sigmoid



(b) tanh



(c) linear

Figure 2.11: Neural network transfer functions.

squared errors (SSE)

$$E_D = \frac{1}{2}\sum_{i=1}^{N}(y_i - z_i)^2 \tag{2.33}$$

where z_i is the predicted network output for observation i. The backpropagation algorithm originally proposed by Rumelhart et al. is probably the best known example of the above mechanics [197]. It performs the optimization by using repeated evaluation of the gradient of E_D and the chain rule of derivative calculus. Due to the problems of slow convergence and relative inefficiency of this algorithm, new and improved optimization methods (e.g. Levenberg-Marquardt and Quasi-Newton) have been suggested. For an overview, see [26].

It has to be noticed that for classification purposes the sum of squared error function E_D of Equation 2.33 is no longer the most appropriate optimization criterion because it was derived using maximum likelihood reasoning on the assumption of Gaussian distributed target data [26, 35, 218]. Since the target output is categorical in a classification context, this assumption is no longer valid. A more appropriate objective function is the cross-entropy function which was explained in subsection 2.1.1:

$$G = -\sum_{i=1}^{N} \{y_i \log(z_i) + (1 - y_i)\log(1 - z_i)\}. \tag{2.34}$$

It can easily be verified that this error function reaches its minimum when $z_i = y_i$ for all $i = 1, ..., N$. Optimization of G with respect to the weights $\mathbf{W}$ and $\mathbf{v}$ may be carried out by using the optimization algorithms mentioned in [26]. Note that the logistic regression classifier discussed in section 2.1.1 is basically a simplified neural network with only one neuron with a sigmoid activation function and trained to minimize the cross-entropy error.

The ultimate goal of NN training is to produce a model which performs well on new, unseen test instances. If this is the case, we say that the network generalizes well. To do so, we basically have to avoid the network from fitting the noise or idiosyncracies in the training data. This can be realized by monitoring the error on a separate validation set during training of the network. When the error measure on the validation set starts to increase, training is stopped, thus effectively preventing the network from fitting the noise in the training data (*early stopping*). However, this causes loss of data that cannot be used for estimating the weights and hence, this method may not be appropriate for small data sets.

A superior alternative is to add a penalty term to the objective function as follows [26, 226]

$$F(\mathbf{w}) = G + \alpha E_W \tag{2.35}$$

with

$$E_W = \frac{1}{2}\sum_i \mathbf{w}_i^2 \tag{2.36}$$

whereby $\mathbf{w}$ is the weight vector representing all weights $\mathbf{W}$ and $\mathbf{v}$. This method for improving generalization constrains the size of the network weights and is referred to as regularization. When the weights are kept small, the network response will be smooth. This decreases the tendency of the network to fit the noise in the training data.

The success of NNs with weight regularization obviously depends strongly on finding appropriate values for the weights $\mathbf{w}$ and the hyperparameter α. The Bayesian evidence framework developed by David MacKay [154, 155] allows to set the regularization parameter α on-line during the optimization process. Furthermore, the Automatic Relevance Determination (ARD) extension of this framework allows one to assess the importance of the inputs. Preliminary results have shown that this framework works very well for customer retention scoring [15, 254].

2.1.10 Support Vector Machine Classifiers

Given a training set of N data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, with input data $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{-1, +1\}$, the support vector machine (SVM) classifier, according to Vapnik's original formulation satisfies the following conditions [29, 54, 200, 250, 251, 252]:

$$\begin{cases} \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b \geq +1, & \text{if } \; y_i = +1 \\ \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b \leq -1, & \text{if } \; y_i = -1 \end{cases} \tag{2.37}$$

which is equivalent to

$$y_i[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b] \geq 1, \qquad i = 1, ..., N. \tag{2.38}$$

The non-linear function $\boldsymbol{\varphi}(\cdot)$ maps the input space to a high (possibly infinite) dimensional feature space. In this feature space, the above inequalities basically construct a hyperplane $\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b = 0$ discriminating between both classes. This is visualized in Figure 2.12 for a typical two-dimensional case.

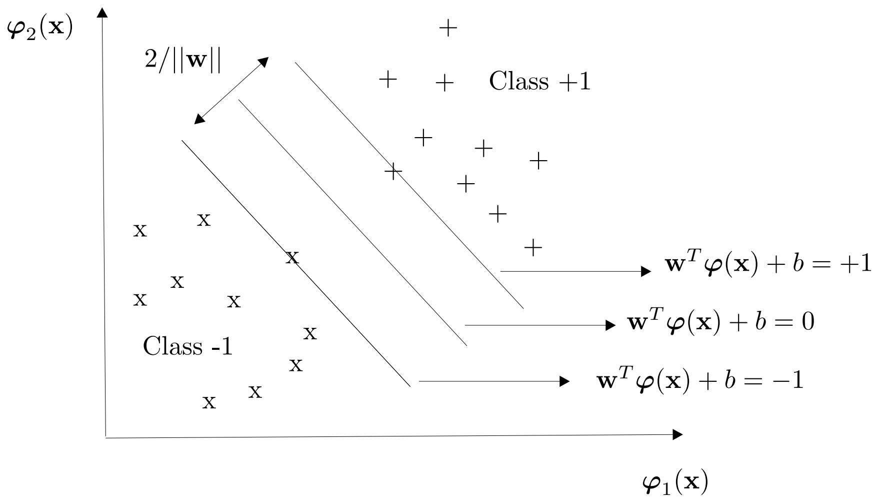

$\boldsymbol{\varphi}_2(\mathbf{x})$
$2/||\mathbf{w}||$
Class +1
Class -1
$\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b = +1$
$\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b = 0$
$\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b = -1$
$\boldsymbol{\varphi}_1(\mathbf{x})$

Figure 2.12: Illustration of SVM optimization of the margin in the feature space.

In primal weight space the classifier then takes the form

$$y(\mathbf{x}) = \text{sign}[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b], \tag{2.39}$$

but, on the other hand, is never evaluated in this form. One defines the convex optimization problem:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{J}(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^N \xi_i \tag{2.40}$$

subject to

$$\begin{cases} y_i[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i)+b] \geq 1-\xi_i, & i=1,...,N \\ \xi_i \geq 0, & i=1,...,N. \end{cases} \tag{2.41}$$

The variables ξ_i are slack variables which are needed in order to allow misclassifications in the set of inequalities (e.g. due to overlapping distributions). The first part of the objective function tries to maximize the margin between both classes in the feature space, whereas the second part minimizes the misclassification error. The positive real constant C should be considered as a tuning parameter in the algorithm. Notice that this formulation is closely related to the LP formulation. The major differences are that the SVM classifier introduces a large margin (or regularization) term $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ in the objective function, considers a margin to separate the classes, and allows for non-linear decision boundaries because of the mapping $\boldsymbol{\varphi}(\cdot)$.

The Lagrangian to the constraint optimization problem (2.40) and (2.41) is given by

$$\mathcal{L}(\mathbf{w},b,\boldsymbol{\xi};\boldsymbol{\alpha},\boldsymbol{\nu}) = \mathcal{J}(\mathbf{w},b,\boldsymbol{\xi}) - \sum_{i=1}^{N}\alpha_i\{y_i[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i)+b]-1+\xi_i\} - \sum_{i=1}^{N}\nu_i\xi_i. \tag{2.42}$$

The solution to the above optimization problem is given by the saddle point of the Lagrangian, i.e. by minimizing $\mathcal{L}(\mathbf{w},b,\boldsymbol{\xi};\boldsymbol{\alpha},\boldsymbol{\nu})$ with respect to $\mathbf{w}$, b, $\boldsymbol{\xi}$ and maximizing it with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\nu}$:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\nu}} \min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{L}(\mathbf{w},b,\boldsymbol{\xi};\boldsymbol{\alpha},\boldsymbol{\nu}). \tag{2.43}$$

One obtains

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial\mathbf{w}} = 0 & \rightarrow & \mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i\boldsymbol{\varphi}(\mathbf{x}_i) \\ \frac{\partial\mathcal{L}}{\partial b} = 0 & \rightarrow & \sum_{i=1}^{N}\alpha_i y_i = 0 \\ \frac{\partial\mathcal{L}}{\partial\xi_i} = 0 & \rightarrow & 0 \leq \alpha_i \leq C \quad , \quad i=1,...,N. \end{cases} \tag{2.44}$$

By substituting the first expression into (2.39), the resulting classifier now becomes:

$$y(\mathbf{x}) = \text{sign}[\sum_{i=1}^{N}\alpha_i\, y_i\, K(\mathbf{x}_i,\mathbf{x})+b], \tag{2.45}$$

whereby $K(\mathbf{x}_i,\mathbf{x}) = \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x})$ is taken with a positive definite kernel satisfying the Mercer theorem.

The Lagrange multipliers α_i are then determined by means of the following optimization problem (dual problem):

$$\max_{\alpha_i} -\frac{1}{2}\sum_{i,j=1}^{N} y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)\alpha_i\alpha_j + \sum_{i=1}^{N}\alpha_i \tag{2.46}$$

subject to

$$\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C, \quad i = 1, ..., N. \end{cases} \tag{2.47}$$

The entire classifier construction problem now simplifies to a convex quadratic programming (QP) problem in α_i. Note that one does not have to calculate $\mathbf{w}$ nor $\boldsymbol{\varphi}(\mathbf{x}_i)$ in order to determine the decision surface. Thus, no explicit construction of the nonlinear mapping $\boldsymbol{\varphi}(\mathbf{x})$ is needed. Instead, the kernel function K will be used. For the kernel function $K(\cdot,\cdot)$ one typically has the following choices:

$K(\mathbf{x},\mathbf{x}_i) = \mathbf{x}_i^T\mathbf{x}$,	(linear kernel)
$K(\mathbf{x},\mathbf{x}_i) = (1+\mathbf{x}_i^T\mathbf{x}/c)^d$,	(polynomial kernel of degree d)
$K(\mathbf{x},\mathbf{x}_i) = \exp\{-\|\mathbf{x}-\mathbf{x}_i\|_2^2/\sigma^2\}$,	(RBF kernel)
$K(\mathbf{x},\mathbf{x}_i) = \tanh(\kappa\,\mathbf{x}_i^T\mathbf{x}+\theta)$,	(MLP kernel),

where d, c, σ, κ and θ are constants. Notice that the Mercer condition holds for all $c, \sigma \in \mathbb{R}^+$ and $d \in \mathbb{N}$ values in the polynomial and RBF case, but not for all possible choices of κ and θ in the MLP case. The scale parameters c, σ and κ determine the scaling of the inputs in the polynomial, RBF and MLP kernel function. This scaling is related to the bandwidth of the kernel in statistics, where it is shown that the bandwidth is an important parameter of the generalization behavior of a kernel method [188].

Typically, many of the α_i will be equal to zero (sparseness property). The training observations corresponding to non-zero α_i are called support vectors and are located close to the decision boundary.

2.1.11 Least Squares Support Vector Machine Classifiers

Vapnik's SVM classifier formulation was modified in [227] into the following LS-SVM formulation:

$$\min_{\mathbf{w},b,\mathbf{e}} \mathcal{J}(\mathbf{w},b,\mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \gamma\,\frac{1}{2}\sum_{i=1}^{N} e_i^2 \tag{2.48}$$

subject to the equality constraints

$$y_i\,[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i)+b] = 1-e_i,\ \ i=1,...,N. \tag{2.49}$$

This formulation consists of equality instead of inequality constraints and takes into account a squared error with regularization term similar to ridge regression.

The solution is obtained after constructing the Lagrangian:

$$\mathcal{L}(\mathbf{w},b,\mathbf{e};\boldsymbol{\alpha}) = \mathcal{J}(\mathbf{w},b,\mathbf{e}) - \sum_{i=1}^{N}\alpha_i\{y_i[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i)+b]-1+e_i\}, \tag{2.50}$$

where $\alpha_i \in \mathbb{R}$ are the Lagrange multipliers that can be positive or negative in the LS-SVM formulation. From the conditions for optimality, one obtains the Karush-Kuhn-Tucker (KKT) system:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 & \rightarrow & \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{\varphi}(\mathbf{x}_i) & \\ \frac{\partial \mathcal{L}}{\partial b} = 0 & \rightarrow & \sum_{i=1}^{N} \alpha_i y_i = 0 & \\ \frac{\partial \mathcal{L}}{\partial e_i} = 0 & \rightarrow & \alpha_i = \gamma e_i, & i = 1, ..., N \\ \frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 & \rightarrow & y_i[\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b] - 1 + e_i = 0, & i = 1, ..., N. \end{cases} \tag{2.51}$$

Note that sparseness is lost which is clear from the condition $\alpha_i = \gamma e_i$. As in standard SVMs, we never calculate $\mathbf{w}$ nor $\boldsymbol{\varphi}(\mathbf{x}_i)$. Therefore, we eliminate $\mathbf{w}$ and $\mathbf{e}$ yielding [224]

$$\left[\begin{array}{c|c} 0 & \mathbf{y}^T \\ \hline \mathbf{y} & \boldsymbol{\Omega} + \gamma^{-1}\mathbf{I} \end{array}\right] \left[\begin{array}{c} b \\ \hline \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{1} \end{array}\right] \tag{2.52}$$

with $\mathbf{y} = [y_1; ...; y_N]$, $\mathbf{1} = [1; ...; 1]$, $\mathbf{e} = [e_1; ...; e_N]$, $\boldsymbol{\alpha} = [\alpha_1; ...; \alpha_N]$. Mercer's condition is applied within the $\boldsymbol{\Omega}$ matrix

$$\boldsymbol{\Omega}_{ij} = y_i y_j \, \boldsymbol{\varphi}(\mathbf{x}_i)^T \boldsymbol{\varphi}(\mathbf{x}_j) = y_i y_j \, K(\mathbf{x}_i, \mathbf{x}_j). \tag{2.53}$$

The LS-SVM classifier is then constructed as follows:

$$y(\mathbf{x}) = \text{sign}[\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b]. \tag{2.54}$$

Note that the matrix in (2.52) is of dimension $(N + 1) \times (N + 1)$. For large values of N, this matrix cannot easily be stored, such that an iterative solution method for solving it is needed. A Hestenes-Stiefel conjugate gradient algorithm is suggested in [224] to overcome this problem. Basically, the latter rests upon a transformation of the matrix in (2.52) to a positive definite form [224]. A straightforward extension of LS-SVMs to multiclass problems has been proposed in [228], where additional outputs are taken in order to encode multiple classes as is often done in classical neural networks methodology [26]. A drawback of LS-SVMs is that sparseness is lost due the choice of a 2-norm. However, this can be circumvented in a second stage by a pruning procedure which is based upon removing training points guided by the sorted support value spectrum [225]. In [242, 244], the LS-SVM classifier formulation was related to regularized Fisher discriminant analysis in the feature space and the use of Fisher targets $\{-N/N_-, +N/N_+\}$ as an alternative to the targets $\{-1, +1\}$ was presented and discussed (see also [92, 240] for a more elaborate discussion).

2.2 Data Set Split Up

Assessing the predictive power of a classification technique is by no means a trivial exercise. One could use the same data for both training and estimating the accu-

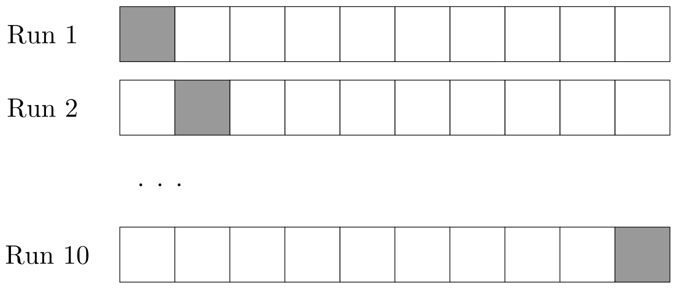


Figure 2.13: 10-fold cross-validation.

racy of a classifier (*resubstitution estimate*). However, this will often result in an overly optimistic accuracy estimate since a trained classifier is typically somewhat biased towards the training data. This is especially the case for classification techniques with many parameters (e.g. neural networks) which are rather sensitive to overfitting i.e. modeling the noise in the training data. Note however that criteria have been suggested in the literature that penalize the training set performance according to the complexity of the classification technique and thus to its ability of overfitting. Examples are the Akaike Information Criterion [2], the Generalized Prediction Error [164] and the Network Information Criterion [166].

Another more popular method for performance assessment is the k-fold cross-validation (CV) method whereby the data set is split into k mutually exclusive folds of nearly equal size [139]. The classifier is then trained k times each time using $k-1$ folds for training (*training folds*) and the remaining fold for evaluation (*validation fold*). The CV performance estimate is then obtained by averaging the k validation fold estimates found during the k runs of the CV procedure. The variance of this estimate can then also be easily computed. If k equals the sample size, this is called *leave-one-out cross-validation.* Common values for k are 5 and 10 [139]. In a stratified cross-validation experiment, all folds have approximately the same class proportions as the original data set. The CV method is often used to assess the performance of classification techniques on small data sets. Figure 2.13 illustrates a 10-fold CV experiment whereby the white squares represent the training folds and the grey squares the validation fold.

For large data sets, one commonly adopts a single training set/test set split up. The classifier is then estimated on the training set and evaluated on the test set. One typically uses $\frac{2}{3}$ of the observations for training and the remaining $\frac{1}{3}$ for testing. In order to obtain a variance of the performance estimate, several sampling strategies may be used. One can create B new data subsamples of predefined size by each time randomly drawing observations with replacement from the original data set (*bootstrapping*)[71]. Each of the B subsamples may then be split into a training set and a test set and the average performance estimate is then found by averaging all B test set performances. Another option is to create a number of randomizations by simply randomizing the order of the observations and then splitting each randomization in a training set and test set (cf. infra).

2.3 Input Selection

Input selection is a commonly adhered technique to reduce model complexity. In a classification context, input selection aims at removing irrelevant or redundant inputs from the classifier. The removal of inputs will lead to faster training and faster evaluation of the classifier. Furthermore, classification models with fewer inputs are also more attractive for humans since they are less complex and thus easier to comprehend. Additionally, input pruning may also augment the predictive power of the classifier [26].

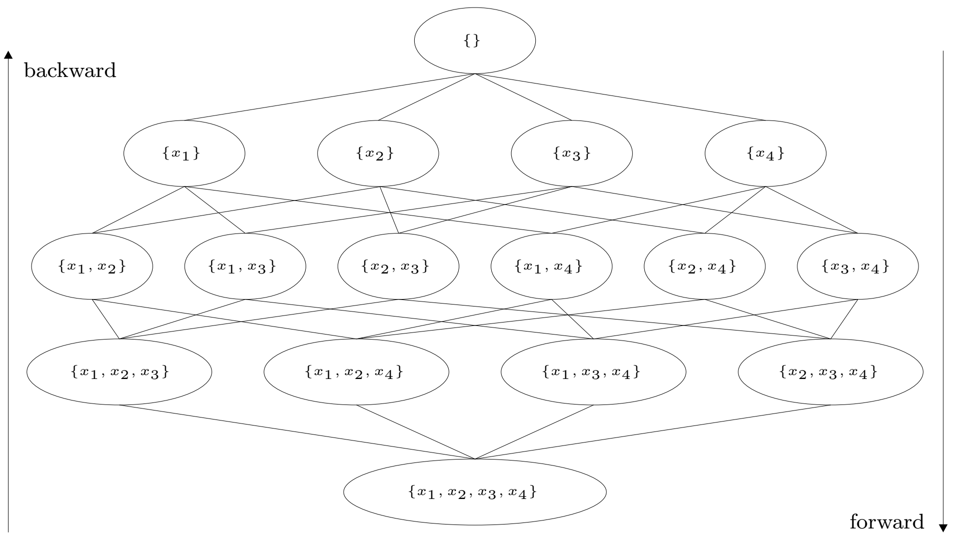


Figure 2.14: Input search space.

Selecting the best subset of a set of n input variables as predictors for a classifier is a non-trivial problem. This follows from the fact that the optimal input subset can only be obtained when the input space is exhaustively searched. When n inputs are present, this would imply the need to evaluate $2^n - 1$ input subsets (see Figure 2.14). Unfortunately, as n grows, this very quickly becomes computationally infeasible [129]. For that reason, heuristic search procedures are often preferred. The *backward selection* scheme starts from a full input set and step-wise prunes input variables that are undesirable according to some heuristic. The *forward selection* scheme starts from the empty input set and step-wise adds input variables that are desirable.

Input selection can either be performed as a pre-processing step, independent of the classification algorithm, or explicitly make use of it. The former approach is termed *filter*, the latter *wrapper* [18, 129]. Filter methods operate independently of any learning algorithm. Undesirable inputs are filtered out of the data before the classifier is trained. Filters typically make use of correlation based or entropy based measures to detect the redundant inputs. Among the well-known filter approaches are FOCUS [4] and Relief [138]. Wrapper methods make use of the trained classifier to evaluate the usefulness of inputs. The input evaluation heuristic that is used is typically based upon inspection of the trained parameters and/or comparison of the predictive performance under different input subset configurations. Input selection is then often performed in a sequential way, e.g. guided by a best-first input selection strategy. A multitude of wrapper input selection methods have been proposed in the context of neural networks e.g. [15, 18, 28, 164, 190, 191, 209, 254]. These methods generally rely on the use of sensitivity heuristics, which try to measure the impact of input changes on the output of the trained network. Inputs may then be ranked (*soft input selection*) and/or pruned (*hard input selection*) according to their sensitivity values.

Example 2.5
In [254], we studied the problem of repeat-purchase modeling in a direct marketing setting using Belgian data. The case involved the detection and qualification of the most relevant RFM (Recency, Frequency and Monetary) variables for predicting purchase incidence. We hereby used a neural network wrapper as our input pruning method. Results indicated that the elimination of redundant and/or irrelevant inputs by means of the discussed input selection method allowed us to significantly reduce model complexity without degrading the predictive generalization ability. This research was further extended in [255], where a wrapper was constructed using an LS-SVM classifier. The proposed wrapper also performed very well for the case at hand.

Example 2.6
In [15], we used the evidence framework of MacKay to perform soft input selection for the same direct marketing case as in Example 2.5. The automatic relevance determination (ARD) method, an integrated feature of this framework, allowed us to assess the relative importance of the inputs in a straightforward and integrated way.

Example 2.7
In [16], an initial approach to wrapped input selection using LS-SVM classifiers was presented and evaluated on five publicly available real-life benchmark UCI data sets. The results indicated that for the majority of the discussed data sets, the model complexity could be substantially reduced.

Example 2.8
In [241], we applied a wrapped backward input selection scheme using LDA, logistic regression and LS-SVM classifiers for predicting bankruptcy of mid-cap firms in Belgium and the Netherlands. It was shown that the LS-SVM classifier, trained on the reduced input set, performed very well when compared to the pruned LDA and logistic regression classifiers.

Note also that feature construction methods are sometimes used as a preprocessing step before the classifier is trained. These methods generally try to construct new features based on the original inputs. One popular example is principal component analysis which aims at transforming the original (possibly) correlated inputs into a (smaller) number of uncorrelated inputs called principal components [131]. However, this dimensionality reduction involves the loss of some information, and care should be taken that the lost information is not crucial for the classification task [26].

2.4 Example Benchmarking Study 1

In this section, we present an example benchmarking study involving some of the classification techniques discussed in the previous subsections on 10 benchmark binary classification data sets [27], of which a brief description will be included in the next subsection.

2.4.1 Description of the Data Sets

All data sets have been obtained from the publicly accessible UCI benchmark repository [27] at `http://kdd.ics.uci.edu/`. These data sets have been referred to numerous times in the literature, which makes them very suitable for benchmarking purposes. As a preprocessing step, all observations containing unknown values are removed from consideration. The following binary data sets were retrieved from [27]: the Statlog Australian credit (`acr`), the Bupa liver disorders (`bld`), the Statlog German credit (`gcr`), the Statlog heart disease (`hea`), the Johns Hopkins university ionosphere (`ion`), the Pima Indians diabetes (`pid`), the sonar (`snr`), the tic-tac-toe endgame (`ttt`), the Wisconsin breast cancer (`wbc`) and the adult (`adu`) data set. The main characteristics of these data sets are summarized in Table 2.1. Note that in Table 2.1, N_{CV} stands for the number of data points

	`acr`	`bld`	`gcr`	`hea`	`ion`	`pid`	`snr`	`ttt`	`wbc`	`adu`
N_{CV}	460	230	666	180	234	512	138	638	455	33000
N_{test}	230	115	334	90	117	256	70	320	228	12222
N	690	345	1000	270	351	768	208	958	683	45222
n_{num}	6	6	7	7	33	8	60	0	9	6
n_{cat}	8	0	13	6	0	0	0	9	0	8
n	14	6	20	13	33	8	60	9	9	14

Table 2.1: Characteristics of the binary classification UCI data sets.

used in the cross-validation based tuning procedure, N_{test} for the number of observations in the test set (see next subsection) and N for the total data set size. The number of numerical and categorical attributes is denoted by n_{num} and n_{cat} respectively, n is the total number of attributes.

2.4.2 Hyperparameter Selection for the SVM Classifiers

Different techniques exist for tuning the hyperparameters related to the regularization constant and the parameter of the kernel function of the LS-SVM and SVM classifier. Among the available tuning methods we find minimization of the Vapnik-Chervonenkis (VC) dimension [26, 216, 229, 250, 252], the use of cross-validation methods, bootstrapping techniques, Bayesian inference [26, 142, 154, 243, 244], etc. Here, we will select the regularization and kernel parameters of both the LS-SVM and SVM classifier using a simple 10-fold cross-validation procedure.

In the case of an RBF kernel, the hyperparameter γ, the kernel parameter σ and the test set performance of the LS-SVM classifier are estimated using the following steps:

1. Set aside 2/3 of the data for the training/validation set and the remaining 1/3 for testing.
2. Starting from $i = 0$, perform 10-fold cross-validation on the training/validation data for each (σ, γ) combination from the initial candidate tuning sets[6] $\Sigma_0 = \{0.5, 5, 10, 15, 25, 50, 100, 250, 500\} \cdot \sqrt{n}$ and $\Gamma_0 = \{0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000\}$.
3. Choose optimal (σ, γ) from the tuning sets Σ_i and Γ_i by looking at the best cross-validation performance for each (σ, γ) combination.
4. If $i = i_{max}$, go to step 5; else $i := i + 1$, construct a locally refined grid $\Sigma_i \times \Gamma_i$ around the optimal hyperparameters (σ, γ) and go to step 3.
5. Construct the LS-SVM classifier using the total training/validation set for the optimal choice of the tuned hyperparameters (σ, γ).
6. Assess the test set accuracy by means of the independent test set.

In this benchmarking study, i_{max} was typically set to 3. This involves a fine-tuned selection of both the σ and γ parameters. It should be remarked however that the refining of the grid is not always necessary as the 10-fold (CV10) cross-validation performance typically has a flat maximum, as can be seen from Figure 2.15 for the `ion` data set. The function depicted in Figure 2.15 is rather flat near the maximum. The CV10 accuracy is more sensitive to the kernel or bandwidth parameter σ

[6]The square root $\sqrt{n}$ of the number of inputs n is considered in the grid Σ (Step 2) since $||\mathbf{x} - \mathbf{x}_i||_2^2$ in the RBF kernel is proportional to n.

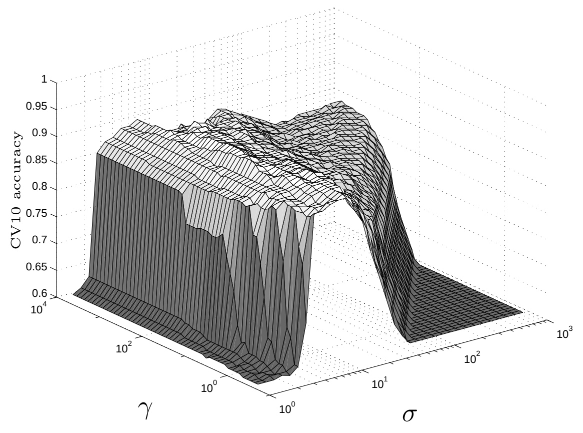


Figure 2.15: Cross-validation (CV10) classification accuracy on the `ion` data set as a function of the regularization parameter γ and kernel parameter σ for an LS-SVM classifier with RBF kernel.

selection [188] than to the choice of the regularization parameter for the `ion` data set.

For the polynomial kernel functions the hyperparameters γ and c were tuned by a similar procedure, while the γ parameter of the linear kernel was selected from a refined set Γ based upon the cross-validation performance.

2.4.3 Experimental Setup

We included the following techniques: the LS-SVM classifier with RBF, linear and polynomial kernel, the SVM classifier with linear and RBF kernel, the C4.5 decision tree induction algorithm, linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), Holte's one rule classifier (oneR), logistic regression (logit), k-nearest neighbor (KNN), and naive Bayes. Although the study could have included still other classification techniques, we believe we consider the most important ones, each originating from different machine learning backgrounds. Note that the oneR classifier is basically a 1-level decision tree using only 1 attribute to classify an observation (see [122] for more details). For the LS-SVM classifier, we will include both the performance of the usual LS-SVM targets $\{-1, +1\}$ and the performance of Regularized Kernel Fisher Discriminant Analysis (LS-SVM$_\mathrm{F}$) targets $\{-N/N_-, +N/N_+\}$. All given n inputs are normalized to zero mean and unit variance in the following way[26]:

$$x_i^{norm} = \frac{x_i - \overline{x_i}}{s_i}, \tag{2.55}$$

whereby $\overline{x_i}$ represents the mean of variable x_i and s_i its sample standard deviation.

The LS-SVMlab Matlab toolbox [177] is used to train and evaluate the LS-SVM classifiers whereas the Matlab SVM toolbox [43] with SMO solver [180] is used to train and evaluate the Vapnik SVM classifier. The C4.5, KNN1, KNN10, naive Bayes and oneR algorithms are implemented using the Weka workbench [263], while the Discriminant Analysis Toolbox (M. Kiefte) for Matlab is applied for LDA, QDA and logit. All experiments are carried out on Sun Ultra5 Workstations and on Pentium II and III PCs.

The oneR, LDA, QDA, logit, NBk and NBn require no parameter tuning. For C4.5, we use the default confidence level of 25% for pruning, which is the value that is commonly used in the machine learning literature. We also experimented with other pruning levels on some of the data sets, but found no significant performance increases. For KNN we use both 1 (KNN1) and 10 (KNN10) nearest neighbors. We use both standard naive Bayes with the normal approximation ($\mathrm{NB_n}$) and the kernel approximation ($\mathrm{NB_k}$) for continuous attributes. The default classifier or majority rule (Maj. Rule) is also included as a baseline in the comparison tables. All comparisons are made on the same randomizations. For another comparison study among 22 decision tree, 9 statistical and 2 neural network algorithms, we refer to [151].

The comparison is performed on an out-of-sample test set consisting of 1/3 of the data. The first 2/3 of each data set is reserved for training and/or cross-validation. For each algorithm, we report the average test set performance and sample standard deviation on 10 randomizations of each data set (see Table 2.4). The best average test set performance is underlined and denoted in bold face for each data set. We then use a paired t-test to test the performance differences:

$$t = \frac{\overline{d}}{\frac{s_d}{\sqrt{10}}} \quad \text{with 9 degrees of freedom}, \tag{2.56}$$

whereby $\overline{d}$ represents the mean performance difference between two classifiers and s_d the corresponding standard deviation. Performances that are not significantly different at the 5% level from the top performance with respect to a one-tailed paired t-test are tabulated in bold face. Statistically significant underperformances at the 1% level are emphasized in italics. Performances significantly different at the 5% level but not a the 1% level are reported in normal script. Since the observations of the randomizations are not independent [63], we remark that this standard t-test is used as a common heuristic to test the performance differences. Ranks are also assigned to each algorithm starting from 1 for the best average performance and ending with 18 for the algorithm with worst performance. Averaging over all data sets, the Average Accuracy (AA) and Average Rank (AR) are reported for each algorithm [151]. A Wilcoxon signed rank test of equality of medians is used on both AA and AR to check whether the performance of an algorithm is significantly different from the algorithm with the highest accuracy

[213]. A Probability of a Sign Test (P_{ST}) is also reported comparing each algorithm to the algorithm with best accuracy [213]. The results of these significance tests on the average data set performances are denoted in the same way as the performances on each individual data set.

2.4.4 Results

LS-SVM	`acr`	`bld`	`gcr`	`hea`	`ion`	`pid`	`snr`	`ttt`	`wbc`	`adu`
N_{CV}	460	230	666	180	234	512	138	638	455	33000
n	14	6	20	13	33	8	60	9	9	14
RBF	0.86	0.72	0.76	**0.83**	**0.96**	0.78	0.77	0.99	**0.97**	**0.85**
Lin	0.86	0.67	0.74	0.83	0.87	**0.78**	0.78	0.66	0.96	0.82
Pol $d = 2$	0.86	0.72	0.76	0.83	0.91	0.78	**0.82**	0.98	0.97	0.84
Pol $d = 3$	0.86	**0.73**	0.76	0.83	0.91	0.78	0.82	0.99	0.97	0.84
Pol $d = 4$	0.86	0.72	0.77	0.83	0.78	0.78	0.81	**1.00**	0.97	0.84
Pol $d = 5$	**0.87**	0.72	0.76	0.83	0.78	0.78	0.81	**1.00**	0.97	0.84
Pol $d = 6$	0.86	0.73	**0.77**	0.83	0.78	0.78	0.81	**1.00**	0.97	0.84
Pol $d = 7$	0.86	0.72	0.77	0.83	0.78	0.78	0.81	**1.00**	0.97	0.84
Pol $d = 8$	0.86	0.73	0.76	0.83	0.78	0.78	0.81	**1.00**	0.97	0.84
Pol $d = 9$	0.86	0.73	0.77	0.83	0.78	0.78	0.81	0.99	0.97	0.84
Pol $d = 10$	0.86	0.71	0.77	0.83	0.91	0.78	0.81	**1.00**	0.97	0.84

Table 2.2: Validation set performance of LS-SVMs on 10 data sets, the best performances on each data set are underlined and denoted in bold face. Performances are represented as percentages.

Tables 2.2 and 2.3 depict the validation and test set performances of the LS-SVM classifiers using the various kernel types on the 10 data sets. The corresponding training set performances are given in Table A.1 of the Appendix. The best validation and test set performances are underlined and denoted in bold face. These experimental results indicate that the RBF kernel yields the best validation and test set performance, while also polynomial kernels yield good performances. These results confirm the results found in [17]. Note that we also conducted the analyses using non-scaled polynomial kernels, i.e., with $c = 1$. For this scaling parameter, LS-SVMs with polynomial kernels of degrees $d = 2$ and $d = 10$ yielded on all data sets average test set performances of 84.3% and 65.9%, respectively. Comparing this with the average test set performance of 85.6% and 85.5% (see Table 2.3 and average the rows corresponding to the polynomial kernels) obtained when using scaling, this clearly motivates the use of bandwidth or kernel parameters. This is especially important for polynomial kernels with degree $d \geq 5$.

Table A.2 of the Appendix presents the optimized values of the regularization parameter γ, and the kernel parameters σ, and c of the LS-SVM classifier with linear, RBF and polynomial kernel. The flat maximum pattern of the CV10 clas-

LS-SVM	acr	bld	gcr	hea	ion	pid	snr	ttt	wbc	adu
N_{test}	230	115	334	90	117	256	70	320	228	12222
n	14	6	20	13	33	8	60	9	9	14
RBF	**0.90**	0.71	**0.77**	0.87	**0.97**	0.77	0.76	0.99	**0.96**	**0.84**
Lin	0.90	**0.72**	0.77	0.86	0.88	0.77	0.74	0.67	0.96	0.82
Pol $d = 2$	0.89	0.71	0.76	0.84	0.93	**0.78**	**0.86**	0.99	0.96	0.84
Pol $d = 3$	0.90	0.71	0.77	0.84	0.93	0.77	0.83	0.99	0.96	0.85
Pol $d = 4$	0.89	0.70	0.76	0.86	0.91	0.78	0.83	**1.00**	0.96	0.84
Pol $d = 5$	0.90	0.72	0.75	0.87	0.88	0.76	0.83	1.00	0.96	0.84
Pol $d = 6$	0.89	0.71	0.76	**0.88**	0.91	0.77	0.84	**1.00**	0.96	0.85
Pol $d = 7$	0.89	0.70	0.75	0.87	0.91	0.77	0.79	**1.00**	0.96	0.85
Pol $d = 8$	0.88	0.70	0.76	0.86	0.91	0.76	0.83	**1.00**	0.96	0.85
Pol $d = 9$	0.88	0.70	0.76	0.86	0.90	0.77	0.80	0.99	0.96	0.84
Pol $d = 10$	0.89	0.71	0.75	0.88	0.94	0.78	0.80	**1.00**	0.96	0.84

Table 2.3: Test set performance of LS-SVMs on 10 binary data sets, the best performances on each data set are underlined and denoted in bold face. Performances are represented as percentages.

sification accuracy illustrated in Figure 2.15 for the `ion` data set was commonly encountered among all evaluated data sets.

The regularization parameter C and kernel parameter σ of the SVM classifiers with linear and RBF kernels were selected in a similar way as for the LS-SVM classifier using the 10-fold cross-validation procedure outlined in section 2.4.2. The optimal hyperparameters of the SVM classifiers are reported in Table A.3 of the Appendix.

The optimized regularization and kernel parameters are then used to assess the test set performance of the SVM, LS-SVM and LS-SVM$_{\text{F}}$ classifiers on 10 randomizations of each data set: for each randomization the first 2/3 of the data are used for training, while the remaining 1/3 is set aside for testing. In the same way, the test set performances of the other classification techniques are assessed. The same randomizations are used in order to make a fair comparison between all classifiers possible. Both sample mean and sample standard deviation of the performance on the different data sets are reported in Table 2.4 using the bold, normal and emphasized script to enhance the visual interpretation as explained above. Averaging over all data sets, the mean performance and rank and the probability of different medians with respect to the best algorithm are tabulated in the last 3 columns of the Table.

The LS-SVM classifier with Radial Basis Function kernel (RBF LS-SVM) achieves the best average test set performance on 3 of the 10 benchmark data sets, while its accuracy is not significantly worse than the best algorithm on 3 other data sets. LS-SVM classifiers with polynomial and linear kernel yield the best performance on two and one data set(s), respectively. Also the RBF SVM, KNN1, NB_k and C4.5 classifier achieve the best performance on one data set each. A comparison of the accuracy achieved by the non-linear polynomial and RBF kernel with the accuracy of the linear kernel illustrates that most data sets are only weakly non-linear. The LS-SVM formulation with targets $\{-1, +1\}$ yields a better performance than the LS-SVM_F regression formulation related to regularized kernel Fisher's discriminant with targets $\{-N/N_-, +N/N_+\}$, although not all tests report a significant difference. Noticing that the LS-SVM with linear kernel and without regularization ($\gamma \to \infty$) corresponds to the LDA classifier, we also remark that a comparison of both accuracies indicates that the use of regularization slightly improves the generalization behavior.

Considering the Average Accuracy (AA) and Average Ranking (AR) over all data sets, the RBF SVM gets the best average accuracy and the RBF LS-SVM the best average rank. There is no significant difference between the performance of both classifiers. The average performance of the Pol LS-SVM and Pol LS-SVM_F classifiers is not significantly different with respect to the best algorithms. The performances of many other advanced SVM algorithms are in line with the above results [32, 162, 200]. The significance tests on the average performances of the other classifiers do not always yield the same results. Generally speaking, the performance of the Lin LS-SVM, Lin SVM, Logit, NB_k and KNN1 classifiers is not significantly different from the best performance at the 1% level. Also the performances of LS-SVMs with Fisher's discriminant targets (LS-SVM_F) are not significantly different at the 1% level.

In summary, one may conclude that the SVM and LS-SVM formulations achieve very good test set performances compared to the other reference classification algorithms. However, it has to be noted that simple, linear classifiers, e.g. logistic regression, also achieve very good classification accuracy which indicates that most of the data sets used are only weakly non-linear.

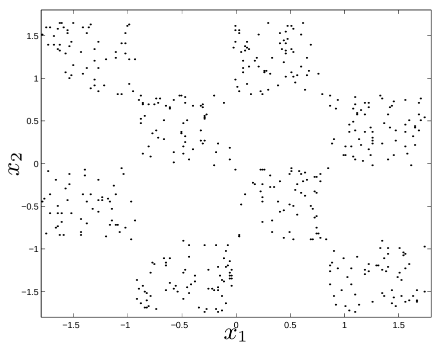
Note that in [242], this study was also extended to a multiclass setting where we considered 10 UCI multiclass data sets and used different output coding schemes for the (LS-)SVM classifiers. We have included the results of this study in Table B.2 of the Appendix.

2.5 Benchmarking Study 2

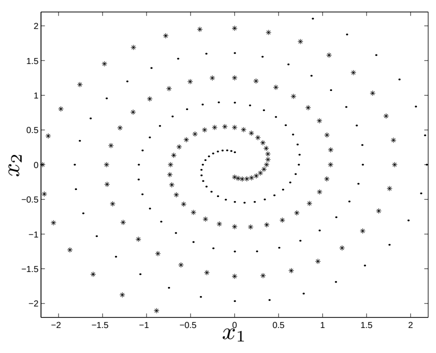
In a second benchmarking study, we will consider two highly non-linear artificial classification problems. The first problem concerns a checkerboard where the aim is to distinguish the white squares from the black squares based on their location. The second problem is the well-known 2-spiral problem which consists of two intertwined spirals [227]. Both classification problems are depicted in Figure 2.16. The data set for the checkerboard has 1000 observations whereas the 2-

	acr	bld	gcr	hea	ion	pid	snr	ttt	wbc	adu	AA	AR	P_{ST}
N_{test}	230	115	334	90	117	256	70	320	228	12222			
n	14	6	20	13	33	8	60	9	9	14			
RBF LS-SVM	$\underline{\mathbf{87.0}}$(2.1)	**70.2**(4.1)	$\underline{\mathbf{76.3}}$(1.4)	**84.7**(4.8)	$\underline{\mathbf{96.0}}$(2.1)	**76.8**(1.7)	*73.1*(4.2)	*99.0*(0.3)	*96.4*(1.0)	*84.7*(0.3)	**84.4**	$\underline{\mathbf{3.5}}$	**0.727**
RBF LS-SVM$_F$	**86.4**(1.9)	*65.1*(2.9)	*70.8*(2.4)	83.2(5.0)	*93.4*(2.7)	*72.9*(2.0)	*73.6*(4.6)	*97.9*(0.7)	96.8(0.7)	*77.6*(1.3)	*81.8*	8.8	**0.109**
Lin LS-SVM	**86.8**(2.2)	*65.6*(3.2)	75.4(2.3)	$\underline{\mathbf{84.9}}$(4.5)	*87.9*(2.0)	76.8(1.8)	*72.6*(3.7)	*66.8*(3.9)	*95.8*(1.0)	*81.8*(0.3)	79.4	7.7	**0.109**
Lin LS-SVM$_F$	**86.5**(2.1)	*61.8*(3.3)	*68.6*(2.3)	*82.8*(4.4)	*85.0*(3.5)	*73.1*(1.7)	73.3(3.4)	*57.6*(1.9)	**96.9**(0.7)	*71.3*(0.3)	*75.7*	12.1	**0.109**
Pol LS-SVM	**86.5**(2.2)	$\underline{\mathbf{70.4}}$(3.7)	**76.3**(1.4)	**83.7**(3.9)	*91.0*(2.5)	**77.0**(1.8)	**76.9**(4.7)	$\underline{\mathbf{99.5}}$(0.5)	*96.4*(0.9)	*84.6*(0.3)	**84.2**	**4.1**	**0.727**
Pol LS-SVM$_F$	**86.6**(2.2)	*65.3*(2.9)	*70.3*(2.3)	*82.4*(4.6)	*91.7*(2.6)	*73.0*(1.8)	**77.3**(2.6)	*98.1*(0.8)	**96.9**(0.7)	*77.9*(0.2)	82.0	**8.2**	**0.344**
RBF SVM	*86.3*(1.8)	**70.4**(3.2)	**75.9**(1.4)	**84.7**(4.8)	95.4(1.7)	$\underline{\mathbf{77.3}}$(2.2)	**75.0**(6.6)	*98.6*(0.5)	*96.4*(1.0)	*84.4*(0.3)	$\underline{\mathbf{84.4}}$	**4.0**	$\underline{\mathbf{1.000}}$
Lin SVM	**86.7**(2.4)	*67.7*(2.6)	75.4(1.7)	*83.2*(4.2)	*87.1*(3.4)	77.0(2.4)	*74.1*(4.2)	*66.2*(3.6)	*96.3*(1.0)	*83.9*(0.2)	*79.8*	**7.5**	0.021
LDA	*85.9*(2.2)	*65.4*(3.2)	**75.9**(2.0)	**83.9**(4.3)	*87.1*(2.3)	76.7(2.0)	*67.9*(4.9)	*68.0*(3.0)	*95.6*(1.1)	*82.2*(0.3)	*78.9*	*9.6*	*0.004*
QDA	*80.1*(1.9)	*62.2*(3.6)	*72.5*(1.4)	*78.4*(4.0)	*90.6*(2.2)	*74.2*(3.3)	*53.6*(7.4)	*75.1*(4.0)	*94.5*(0.6)	*80.7*(0.3)	*76.2*	*12.6*	*0.002*
Logit	**86.8**(2.4)	*66.3*(3.1)	**76.3**(2.1)	*82.9*(4.0)	*86.2*(3.5)	**77.2**(1.8)	*68.4*(5.2)	*68.3*(2.9)	*96.1*(1.0)	*83.7*(0.2)	79.2	7.8	**0.109**
C4.5	*85.5*(2.1)	*63.1*(3.8)	*71.4*(2.0)	*78.0*(4.2)	*90.6*(2.2)	*73.5*(3.0)	*72.1*(2.5)	*84.2*(1.6)	*94.7*(1.0)	$\underline{\mathbf{85.6}}$(0.3)	*79.9*	*10.2*	0.021
oneR	*85.4*(2.1)	*56.3*(4.4)	*66.0*(3.0)	*71.7*(3.6)	*83.6*(4.8)	*71.3*(2.7)	*62.6*(5.5)	*70.7*(1.5)	*91.8*(1.4)	*80.4*(0.3)	*74.0*	*15.5*	*0.002*
KNN1	*81.1*(1.9)	*61.3*(6.2)	*69.3*(2.6)	*74.3*(4.2)	*87.2*(2.8)	*69.6*(2.4)	$\underline{\mathbf{77.7}}$(4.4)	*82.3*(3.3)	*95.3*(1.1)	*78.9*(0.2)	*77.7*	12.5	0.021
KNN10	**86.4**(1.3)	*60.5*(4.4)	*72.6*(1.7)	*80.0*(4.3)	*85.9*(2.5)	*73.6*(2.4)	*69.4*(4.3)	*94.8*(2.0)	*96.4*(1.2)	*82.7*(0.3)	*80.2*	*10.4*	0.039
NB$_k$	*81.4*(1.9)	*63.7*(4.5)	*74.7*(2.1)	83.9(4.5)	*92.1*(2.5)	75.5(1.7)	*71.6*(3.5)	*71.7*(3.1)	$\underline{\mathbf{97.1}}$(0.9)	*84.8*(0.2)	*79.7*	**7.3**	**0.109**
NB$_n$	*76.9*(1.7)	*56.0*(6.9)	74.6(2.8)	**83.8**(4.5)	*82.8*(3.8)	75.1(2.1)	*66.6*(3.2)	*71.7*(3.1)	*95.5*(0.5)	*82.7*(0.2)	*76.6*	*12.3*	*0.002*
Maj. Rule	*56.2*(2.0)	*56.5*(3.1)	*69.7*(2.3)	*56.3*(3.8)	*64.4*(2.9)	*66.8*(2.1)	*54.4*(4.7)	*66.2*(3.6)	*66.2*(2.4)	*75.3*(0.3)	*63.2*	*17.1*	*0.002*

Table 2.4: Comparison of the 10 times randomized test set performance of all classifiers. Best performances are underlined and denoted in bold face, performances not significantly different at the 5% level are denoted in bold face, performances significantly different at the 1% level are emphasized in italics.



(a) checkerboard



(b) 2-spiral

Figure 2.16: The checkerboard and 2-spiral classification problem.

spiral data set consists of 194 observations (97 observations for each spiral). We trained an LS-SVM classifier with $\sigma = 0.5$ and $\gamma = 0.5$ for both data sets and compared its classification accuracy with LDA, QDA and logit (see Table 2.5). As can be seen from Table 2.5, the LS-SVM classifier clearly outperforms the

	Checkerboard	**2-Spiral**
LDA	53.70	50.52
QDA	47.30	50.52
logit	53.70	50.52
LS-SVM	96.70	98.97

Table 2.5: LDA, QDA, logit and LS-SVM classification accuracy on the checkerboard and 2-spiral classification problem.

other classifiers. This is due to the fact that the logistic regression and LDA classifier assume linear decision boundaries whereas the QDA classifier models a quadratic decision boundary. Since both artificial problems are highly non-linear, the modeling capacity of these classifiers is to limited to achieve a good classification accuracy. On the other hand, the kernel trick allows the LS-SVM classifier to model highly complex, non-linear decision boundaries which results in an excellent classification accuracy for both data sets.

We also generated 40000 extra test points for the checkerboard data set and 4000 extra test points for the 2-spiral data set. The classifications made by the LS-SVM classifier are visualized in Figure 2.17. The figure clearly indicates that the LS-SVM classifier has a very good generalization ability on both data sets.

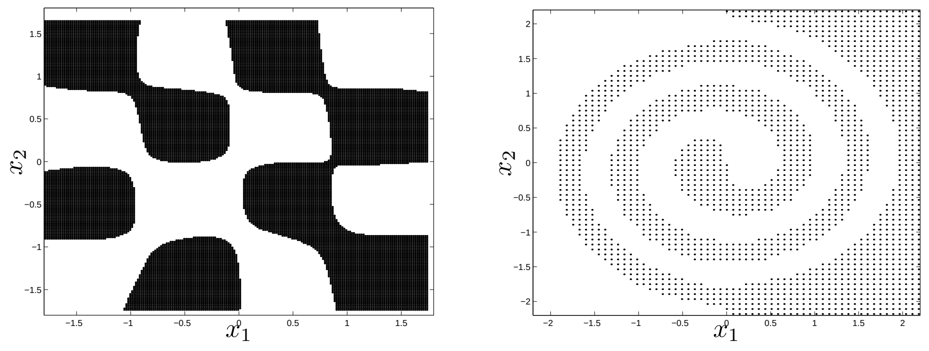


Figure 2.17: LS-SVM generalisation behaviour on the checkerboard and 2-spiral classification problem.

2.6 Conclusions and Limitations of Benchmarking Studies 1 and 2

Benchmarking study 2 clearly illustrated the superiority of LS-SVM classifiers for two artificial, highly non-linear classification problems. However, when looking at Table 2.4 of benchmarking study 1, the performance differences between the non-linear classifiers (LS-SVM and SVM) and simple classifiers e.g. LDA and logit were not that pronounced. The data sets considered in benchmarking study 1 are primarily real-life data sets. This may indicate that patterns such as the checkerboard and 2-spiral pattern depicted in Figure 2.16 are rarely encountered in real-life data sets. Furthermore, it can be argued that most real-life classification data sets can be well separated using simple techniques although non-linear classifiers such as SVMs and LS-SVMs can sometimes provide additional marginal performance benefits. Note that these small performance increases may be very important in a data mining context as the following example illustrates [15, 239].

Example 2.9
Suppose that a mail-order company decides to mail to 75% of its current mailing list of 5 million customers, i.e. $3,750,000$ mailings are sent out. Suppose that the overall response rate when mailing to all of their current customers is 10% during a particular mailing period, i.e. if everyone would be mailed, $500,000$ orders would be placed. Suppose further that the average contribution per customer amounts to 100 Euro, which is the typical real-life situation of a large mail-order company. Table 2.6 compares the economics of several alternative response models each developed using other classification techniques. When no model is available, we can expect to obtain 75% of all potential responses (i.e. $0.75 \times 500,000 = 375,000$ responses) when 75% of 5 million people are mailed (i.e., 3.75 million mailings are sent out). The ideal model (at the specific mailing depth) is able to

Type of Model	Mailing Depth	Customers (million)	Mailings sent out (million)	Responses	Average Cont.	Total Cont.	Additional Cont.
Null Model	75.00 %	5.00	3.75	375,000	100.00	37.50	0.00
Ideal Model	75.00 %	5.00	3.75	500,000	100.00	50.00	12.50
90 % model	75.00 %	5.00	3.75	450,000	100.00	45.00	7.50
91 % model	75.00 %	5.00	3.75	455,000	100.00	45.50	8.00

Table 2.6: Economics resulting from performance differences among response models.

select the people from the mailing list in such a way that the $500,000$ potential customers all receive a mailing, i.e. even though 25% of the mailing list is not mailed, not a single order is lost. Suppose further that the current response model used by the company, by mailing to 75% of their mailing list, allows to obtain 90% of the responses, i.e. even though $1,250,000$ people on the list do not receive a mailing, only 10% of the $500,000$ potential customers are excluded. This will result in $450,000$ orders, which represents a substantial improvement over the 'null model' situation. If a better response model can be built, which achieves 91% of the responses instead of 90%, the contribution of this change will directly increase the contribution over the null model from 7.50 million Euro to 8 million Euro, i.e. by $500,000$ Euro (1% of 10% of 5 million customers $\times 100$ Euro average contribution). Given a tendency of rising mailing costs and increasing competition, we can easily see an increasing importance for developing accurate response models [113]. This example clearly illustrates that an increase of 1% response rate of a direct mailing campaign can result in substantial profit gains.

In benchmarking study 1, we used a one-tailed t-test to test the difference in classification accuracy of two classifiers. However, Dietterich [63] has shown that this test may exhibit somewhat elevated probability of Type I error which is the probability of incorrectly detecting a performance difference when no difference exists.

Another important issue is that both studies used the classification accuracy, measured as the percentage correctly classified (PCC) observations, as the performance measure of interest. Although this measure is undoubtedly the most commonly used, it may not be the most appropriate performance criterion in a number of cases. It tacitly assumes equal misclassification costs for false positive and false negative predictions. This assumption is problematic, since for most real-world problems (e.g. credit scoring, fraud detection) one type of classification error may be much more expensive than the other. Another implicit assumption of the use of PCC as an evaluation metric is that the class distribution (class priors) among the observations is presumed constant over time and relatively balanced [185]. For example, when confronted with a situation characterized by a very skewed class distribution in which faulty predictions for the underrepresented class are very costly, a model evaluated on PCC alone may always predict the most common class and, in terms of PCC, provide a relatively high performance.

Thus, using PCC alone often proves to be inadequate, since class distributions and misclassification costs are rarely uniform. However, taking into account class distributions and misclassification costs proves to be quite hard, since in practice they can rarely be specified precisely and are often subject to change [31, 81, 109]. In spite of the above, comparisons based on classification accuracy often remain useful because they are indicative of a broader notion of good performance [185]. In the following section, we discuss the use of the Area under the Receiver Operating Characteristic Curve as an additional performance measure.

2.7 The Area Under the Receiver Operating Characteristic Curve

Class-wise decomposition of the classification of cases yields a confusion matrix as specified in Table 2.7. The following performance metrics can readily be distilled

	Actual	
Predicted	+	-
+	True Positive (TP)	False Positive (FP)
-	False Negative (FN)	True Negative (TN)

Table 2.7: The confusion matrix for binary classification.

from Table 2.7

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.57}$$

$$\text{specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}}. \tag{2.58}$$

The sensitivity (specificity) measures the proportion of positive (negative) examples which are predicted to be positive (negative). Using the notation of Table 2.7, we may now formulate the overall accuracy as follows

$$\text{PCC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{2.59}$$

Note that sensitivity, specificity and PCC vary together as the threshold on a classifier's continuous output is varied between its extremes within the interval $[0, 1]$. The receiver operating characteristic curve (ROC) is a 2-dimensional graphical illustration of the sensitivity ('true alarms') on the Y-axis versus (1-specificity) ('false alarms') on the X-axis for various values of the classification threshold. It basically illustrates the behavior of a classifier without regard to class distribution or error cost, so it effectively decouples classification performance from these factors [72, 109, 230, 231]. Translated to a credit scoring context, the X-axis depicts the percentage of bads predicted to be good (percentage of goods predicted to be bad) whereas the Y-axis gives the percentage of goods predicted to be good (percentage of bads predicted to be bad).

Figure 2.18 provides an example of several ROC curves. Each ROC curve passes through the points (0,0) and (1,1). The former represents the situation whereby the classification threshold exceeds the highest output posterior probability value (meaning all instances are classified in class 0). In the latter case, the classification threshold is lower than the lowest posterior probability value (meaning all instances are classified in class 1). A straight line through (0,0) and (1,1) represents a classifier found by randomly guessing the class and thus with poor discriminative power, since the sensitivity always equals (1-specificity) for all possible values of the classification threshold (curve A). It is to be considered as a benchmark for the predictive accuracy of other classifiers. The more the ROC curve approaches the (0,1) point, the better the classifier will discriminate (e.g. curve D dominates curves A, B and C). ROC curves of different classifiers may however intersect making a performance comparison less obvious (e.g. curves B and C). To overcome this problem, one often calculates the area under the receiver operating characteristic curve (AUC). The AUC then provides a simple figure-of-merit for the performance of the constructed classifier. An intuitive interpretation of the AUC is that it provides an estimate of the probability that a randomly chosen instance of class 1 (positive instance) is correctly rated (or ranked) higher than a randomly selected instance of class 0 (negative instance). This is equivalent to the Wilcoxon test of ranks [110]. Note that since the area under the diagonal is 0.5, a good classifier should have an AUC larger than 0.5. Since the AUC represents the ability of a classifier to produce good relative instance rankings, it is important to remark that in order to compute the AUC, a classifier need not produce accurate, well calibrated probability estimates. Instead, it need only produce relative accurate scores discriminating the positive and negative instances [80]. Many methods have been suggested to compute the AUC [64, 111]. We will use the well-known trapezoidal rule which calculates the AUC as the sum of the surfaces of a number of adjacent trapezia. It has to be noted however that this method systematically underestimates the true AUC [110]. Although the AUC was originally proposed in the context of binary classification, extensions for multi-class classification have also been recently proposed in the literature [109].

In what follows, we consistently multiply AUC values by a factor of 100 to give a number that is similar to PCC, with 50 indicating random and 100 indicating perfect classification.

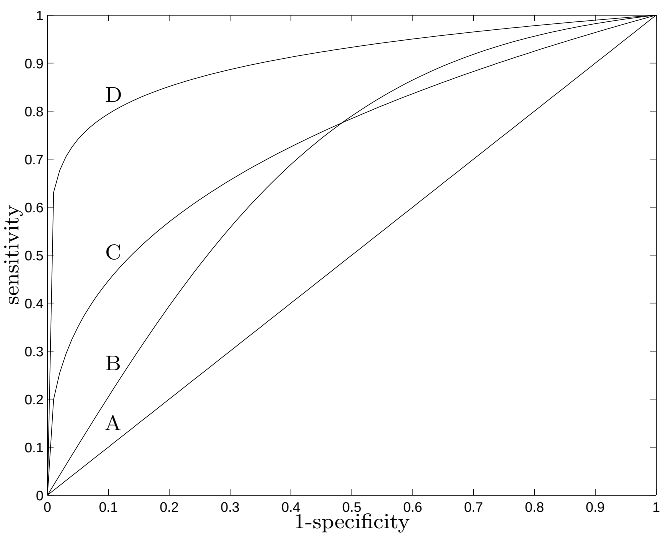


Figure 2.18: The receiver operating characteristic curve (ROC).

2.8 Computing the AUC

For logistic regression, linear and quadratic discriminant analysis, and the naive Bayes classifier the calculation of the AUC poses no problems since each of these classifiers is able to output probabilities of the form $p(y|\mathbf{x})$.

For decision trees and rules, one commonly adopts the confidence of the rule or leave as the class probability estimate. The confidence is defined as $\frac{TP}{TP+FP}$ with $TP(FP)$ the true (false) positives classified by the leave or rule. However, it has to be noted that using the confidence may provide poor probability estimates, especially for rule sets where the rules are no longer mutually exclusive and multiple rules may fire for the same instance. In [80], several voting strategies were compared to compute the AUC of rule sets with overlapping rules. It was shown that for C4.5rules, the strategy of simply using the confidence of the first matching rule provides satisfactorily results. One possible explanation for this is that even after pruning, the number of conflicting rules remains limited. Note that these probability estimates may be further improved by a smoothing operation such as a Laplace correction [184].

For the k-nearest neighbor classifier, the class probability estimate is obtained as the number of votes for the class in question divided by k.

When using logistic transfer functions in the output layer of an MLP, the outputs can be interpreted as class probabilities and the calculation of the AUC becomes straightforward.

For the SVM and LS-SVM classifiers, we use the output of the classifiers before applying the sign operator to compute the AUCs (see Equation 2.54). Remember, in order to compute the AUC, a classifier need not produce accurate probability estimates, but only relative accurate scores discriminating the positive and negative instances (see above). Note however that methods have been proposed in the literature to compute class probability estimates for SVMs [181].

2.9 Test Statistics to Compare the PCC and AUC

Statistically comparing the PCC and AUC values of two classifiers depends on the way both measures have been computed. E.g. when a cross-validation setup was used, one naturally obtains both the mean and standard deviation of the performance criteria and a paired t-test might be used to compare the performance. However, in [63] it was shown that in a cross-validation context, the paired t-test exhibits a somewhat elevated type I error. Since we will use large data sets in the following chapters, we here focus on how to compare the PCC and AUC values based on a training set/test set setup.

To compare the PCC values of two classifiers on the same test set, one commonly adopts the McNemar test [77, 213]. Suppose we have two classifiers, C_1 and C_2, and we represent their misclassifications as the following contingency table:

Number of examples misclassified by both C_1 and C_2 (a)	Number of examples misclassified by C_1 but not by C_2 (b)
Number of examples misclassified by C_2 but not by C_1 (c)	Number of examples misclassified by neither C_1 nor C_2 (d)

Table 2.8: Contingency table for McNemar test.

When conducting the McNemar test, only cells b and c are of interest since they represent the number of misclassifications by C_1 but not by C_2 and vice versa. Under the null hypothesis, both classifiers have the same error rate and b should equal c. However, when classifier C_1 has a higher error rate than classifier C_2, b should be higher than c. When using the notation $\pi_b = \frac{b}{b+c}$ and $\pi_c = \frac{c}{b+c}$ the null and alternative (two tailed) test can be stated as follows [213]:

$$\begin{aligned} H_0 &: \pi_b = \pi_c \\ H_1 &: \pi_b \neq \pi_c. \end{aligned} \tag{2.60}$$

The test statistic for the McNemar test is based on the chi-squared distribution and is computed as follows:

$$\chi^2 = \frac{(b-c)^2}{b+c}, \tag{2.61}$$

with 1 degree of freedom. Note that since the McNemar test employs a continuous distribution to approximate a discrete probability distribution, some authors recommend a correction for continuity as follows

$$\chi^2 = \frac{(|b-c|-1)^2}{b+c}. \tag{2.62}$$

In [63], it was shown that this test has an acceptable Type I error when used in combination with a training set/test set setup.

Multiple tests have been devised to compare AUC values derived from the same test observations. Hanley and McNeil [111] developed a parametric approach which uses a z-statistic based on an approximation procedure that used the Pearson correlation or Kendall tau to estimate the correlation of the two AUCs. DeLong, DeLong and Clarke-Pearson [59] suggested a non-parametric approach whereby the covariance matrix is estimated using the theory on generalized U-statistics. After some mathematical calculus, they arrived at the following test statistic:

$$(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})\mathbf{c}^T[\mathbf{c}\,\mathbf{S}\,\mathbf{c}^T]^{-1}\mathbf{c}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T \tag{2.63}$$

which has a chi-square distribution with degrees of freedom equal to the rank of $\mathbf{c}\,\mathbf{S}\,\mathbf{c}^T$ with $\hat{\boldsymbol{\theta}}$ the vector of the AUC estimates, $\mathbf{S}$ the estimated covariance matrix and $\mathbf{c}$ a vector of coefficients such that $\mathbf{c}\,\boldsymbol{\theta}^T$ represents the desired contrast (see [59] for more details). We will use this test in the remainder of this text.

2.10 Conclusions

In this chapter, we started with discussing a selection of popular classification techniques originating from several backgrounds such as statistics, machine learning and artificial intelligence. We hereby primarily focussed on the basic theoretical underpinnings and aspects relating to the practical implementation of the various classification algorithms. The following techniques were discussed in detail: logistic regression, linear and quadratic discriminant analysis, linear programming, Bayesian networks, C4.5 decision trees and rules, k-nearest neighbor classifiers, neural networks, support vector machines and the recently suggested least squares support vector machine classifiers.

Following the discussion of the classification techniques, we elaborated on ways to split up the data set in order to assess the predictive performance of a classifier. More specifically, we explained the difference between training set and test set, and discussed the well-known k-fold cross-validation procedure in more detail. This was followed by a bird's eye overview on the topic of input selection where a distinction was made between filter methods and wrapper methods. The former basically operate independent of the classification technique whereas the latter are integrated with the classifier using e.g. sensitivity heuristics.

Next, we presented a first benchmarking study which investigated the classification accuracy of the previously discussed classification techniques on 10 publicly available, real-life data sets which have been widely used in the literature. It was concluded that the SVM and LS-SVM classifiers achieved a very good performance although simple classifiers, e.g. logistic regression, also obtained a good classification accuracy, which indicates that the data sets used were only weakly non-linear. In a second benchmarking study, we considered the checkerboard and 2-spiral artificial classification problems which are both very non-linear. It was found that the LS-SVM classifier clearly outperformed the LDA, QDA and logit classifiers by far. When comparing the results of both studies, it was concluded that the performance differences between the non-linear classifiers (e.g. LS-SVM and SVM) and the simple (linear) classifiers (e.g. LDA and logit) were more pronounced in study 2 than in study 1. Hence, our conclusion was that most real-life classification data sets are very good separable using simple classification techniques although non-linear classifiers such as SVMs and LS-SVMs may provide additional performance benefits which may be very important in a data mining context.

One important criticism concerning both benchmarking studies was that they only looked at the classification accuracy of the classification algorithms. It was then argued that misclassification costs and class distributions can also have an impact on the performance. However, since both are hard to specify correctly in advance, other performance measures are needed. We then arrived at the area under the receiver operating characteristic curve as an additional performance metric which measures the performance of a classifier independent of misclassification costs or class distribution. We also discussed how to compute the AUC for the classifiers discussed earlier.

Since knowledge discovery and data mining usually involve the use of large data sets, one commonly adopts a training set/test set procedure to assess the predictive performance of the classifiers. However, when using a single test set, no variance of the performance measure is obtained and appropriate test statistics are required to compare both the classification accuracy and the AUC. We then discussed the McNemar test to compare the classification accuracy and the non-parametric test of DeLong, DeLong and Clarke-Pearson to compare the AUCs.

In the next chapter, we will again evaluate the performance of the classifiers discussed in this chapter on 8 credit scoring data sets, but this time using both the classification accuracy and the AUC as performance measures and the appropriate test statistics to compare them.

Chapter 3

Building Scorecards for Credit Scoring

In this chapter, we will study the topic of building scorecards for credit scoring using the classification techniques discussed in the previous chapter[1,2]*. We will start by discussing the basic problem statement and issues related to credit scoring. It is important to note that we hereby only consider consumer credit scoring instead of corporate credit scoring and bankruptcy prediction which have also deserved a lot of attention in the literature. In a next section, we elaborate on the topic of reject inference which aims at appropriately dealing with the applicants for which credit was denied in the past. This is followed by a literature overview on the use of machine learning techniques for credit scoring.*

In the empirical part of this chapter, we report on the application of the classification techniques discussed in chapter 2 on 8 credit scoring data sets originating from major Benelux (Belgium, The Netherlands and Luxembourg) and U.K. financial institutions. The performance of the trained classifiers is quantified using the classification accuracy and the area under the receiver operating characteristic curve. Various cut-off setting schemes are considered to compute the classification accuracy. Furthermore, aggregate performances and rankings will be computed that will allow us to make some interesting conclusions. A rigorous statistical setup is employed using the appropriate test statistics to compare the performance measures.

[1]B. Baesens, S. Viaene, J. Vanthienen, A Comparative study of state of the art classification algorithms for credit scoring, *Proceedings of the Seventh Conference on Credit Scoring and Credit Control (CSCCVII'2001)*, Edinburgh, Scotland, September, 2001.

[2]B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, J. Vanthienen, Benchmarking State of the Art Classification Algorithms for Credit Scoring, *Journal of the Operational Research Society*, 54(6), pp. 627-635, 2003.

3.1 Problem Statement

In this section, we briefly discuss the basic problem statement and issues related to credit scoring. A more elaborate discussion on this topic can be found in the survey papers by e.g. Hand and Henley [105], Rosenberg and Gleit [196], and Thomas [234], the books published by Hand and Jacka [107] and Thomas et al. [235], and the PhDs of Henley [116] and Kelly [135]. Note that although the term credit scoring is also sometimes used in the context of bankruptcy prediction, we limit ourselves in this text to credit scoring in the context of granting consumer loans.

Credit scoring is a technique that helps organizations decide whether or not to grant credit to consumers who apply to them [234]. These customers may be bank customers borrowing money or retail customers being sold goods on a deferred payment scheme. The primary goal of credit scoring is to summarize all available information about an applicant in a score which reflects the creditworthiness of the applicant. If this score is above a predetermined threshold credit is granted, otherwise credit is denied. The scores are usually computed using scoring algorithms or scorecards which allow one to rank order all applicants by risk.

In the early years of credit scoring, the credit decision was made using a judgmental approach by merely inspecting the application form of the customer. One then commonly focussed on the values of the 5 Cs of a customer: Character, Capital, Collateral, Capacity and Condition [234]. Nowadays, automated scorecards are built using statistical methods to guide the credit granting decision, thereby using both the applicant's application form data and information obtained from credit agencies. Especially logistic regression and discriminant analysis have been widely applied in early credit scoring models [68, 74, 75, 261].

Originally, credit scoring aimed at deciding upon the creditworthiness of new applicants. This is sometimes also referred to as *application scoring*. On the contrary, *behavioral scoring* aims at analyzing the behavior pertaining to the existing customer base. Example questions that are addressed are: Should the bank augment the credit limit of a customer? How should it approach a customer whose financial abilities start deteriorating once a loan was granted? Behavioral scoring is becoming a very popular approach with a lot of potential for monitoring the behavior of the existing customers and undertaking the appropriate actions.

It is important to remark that there is also a legal and ethical issue involved in granting credit to customers. The Equal Credit Opportunities Act (1976) and regulation B in the US prohibit the use of characteristics such as gender, marital status, race, whether an applicant receives welfare payments, color, religion, national origin and age in making the credit decision. In [55], Crook investigated whether organizations generally conform to these principles using a population of households who were discouraged from applying for credit.

A first important issue when building scorecards concerns the definition of a bad customer. Two approaches can be distinguished here. The direct approach constructs a scorecard by choosing a definition of bad on beforehand. Many definitions have been suggested in the literature but the most common definition is that a customer is considered bad if he or she has missed three consecutive months of payments. Some authors even advocate the use of three categories of payers: good payers, bad payers and poor payers [61]. The indirect approach towards scorecard construction first tries to predict the values of other intermediate variables, e.g., balance, excess, months in arrears, which are closely related to the risk of default. The predicted values of these variables are then typically combined using a deterministic model to produce a global risk assessment. The advantage of using the indirect method is that the predicted variables can then also be used to forecast profit. Moreover, one can easily change the definition of bad by simply reformulating the deterministic model without having to re-estimate the prediction model. E.g., Kelly and Hand [136] propose the use of global credit scoring models which allow one to choose the definition of good when the classification is to be made instead of when the scorecard is constructed. Li and Hand [148] develop indirect credit scoring models using a multicriterion definition of bad. They illustrated that after extensive model development the indirect approach gave results which were competitive with those of the direct approach. Figure 3.1 illustrates the difference between the direct and the indirect credit scoring approach.

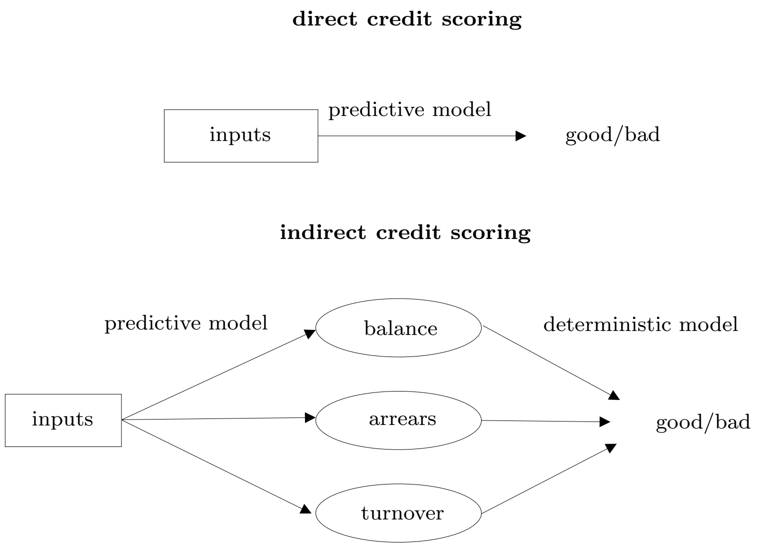


Figure 3.1: Direct versus indirect credit scoring.

Some authors argue to first cluster the population of applicants into homogenous subpopulations with more or less the same risk characteristics and construct a separate scorecard for each subpopulation instead of one overall scorecard built on the entire population of applicants [262]. However, in [20], it was shown that this strategy will not necessarily result in a better discrimination and problems may occur when choosing the appropriate cut-offs for the various individual scorecards.

While most existing credit scoring systems primarily focus on distinguishing bad customers from good customers, it is also very important to know when customers go bad [21, 221]. This information may then be used to e.g calculate the profitability of an individual customer (*profit scoring*) or to improve the debt provisioning strategy of the financial institution. This research question is usually tackled by using survival analysis methods and will be thoroughly investigated in a later chapter.

Scorecards are usually built on a population of past applicants of which the repayment behavior is known in advance. These scorecards are then used to predict the creditworthiness of future applicants. However, since customer populations are likely to change due to e.g. new economic and demographic conditions (*population drift*), the scorecards need to be replaced or dynamically updated at regular periods in time.

The problem of fraud detection is somewhat related to credit scoring in the sense that both share many of the same modeling problems and issues. E.g., in [146], a rule-based expert system is presented to help alert banks and other financial institutions to fraudulent usage of consumer credit cards. In [256], the use of machine learning techniques to predict automobile insurance fraud was investigated.

3.2 Reject Inference

Most scorecards are typically built using a sample of previously accepted loans which turned out to be either bad or good. These scorecards are then subsequently used to score the future population of customers who apply for credit. It is possible that this introduces some bias since a considerable amount of customers for which credit was denied in the past are not properly taken into account in the scorecard development process. Rejected applicants never received credit and their true class label (bad or good customer) will never be known (see Figure 3.2). This problem

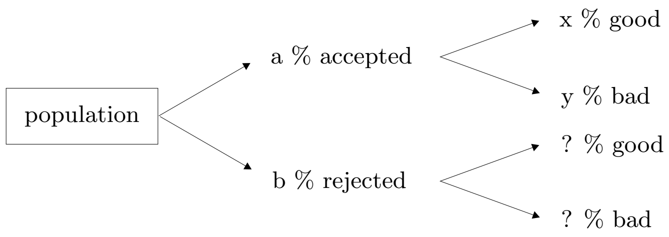


Figure 3.2: The reject inference problem.

has been widely recognized in the credit scoring industry and many attempts have been suggested to take into account the rejected applicants in the scorecard development process. These attempts are commonly referred to as *reject inference*. Reject inference techniques try to assign probabilities of default to the rejected applicants so that they can be included in the construction of the scorecard.

A first approach to reject inference is to consider all rejected applicants as bad and build the scorecard. The main problem with this approach is that it reinforces the screening process that is currently used by the financial institution (see chapter 6 by K. Leonard in the book of Hand and Jacka [107]). Another approach is to look for approved loans which are similar to each rejected loan and assign the class label of the former to the latter. However, the precise determination of 'similar' proves to be quite difficult in practice. One of the most popular methods is to build a scorecard on the accepted applicants and use this scorecard to score and label the rejected applicants as either bad or good. Subsequently, a new scorecard is constructed using the accepted applicants with their observed outcomes and the rejected applicants with their predicted outcomes (see e.g. [220]).

In [104], Hand provides a discussion on the topic of reject inference and concludes his paper by saying:

> *"...there is no unique best method of universal applicability, unless extra information is obtained. That is, the best solution is to obtain more information (perhaps by granting loans to some potential rejects) about those applicants who fall in the reject region [104]".*

Hence, one of the best strategies to deal with reject inference is to grant credit to every applicant during some period of time to gain credit information for the entire population, but of course not many companies are eager to do so because of the amount of credit loss that would be incurred. Nevertheless, in [22] Banasik et al. were in the exceptional situation of being able to observe the repayment behavior of customers that would normally be rejected. They concluded that the scope for improving scorecard performance by including the rejected applicants into the model development process is present but modest.

At present, it can be concluded that there is no consensus upon the need for reject inference and how it should be tackled appropriately. Hence, following Kelly [135], we do not concern ourselves with this extra complication in this thesis. Other papers on this topic are (besides the references given in the introduction of section 3.1) e.g. [85, 101, 128]. In what follows, we will provide a literature overview on the use of machine learning techniques discussed in chapter 2 for credit scoring.

3.3 Literature Overview on using Machine Learning Techniques for Credit Scoring

Feelders et al. [86] compared the performance of linear discriminant analysis and C4.5 decision trees on a data set obtained from the ABN AMRO bank which is a well-known Dutch financial institution. The data set consisted of 2489 observations each described by 38 inputs. The data was split into a training set (1700 Obs.) and a test set (789 Obs.). A customer was categorized as bad if there has been an arrear of 60 days or more. It was found that the p-value of the McNemar test between the linear discriminant and the C4.5 classifier accuracy was 0.1417 and it was left to the reader to decide upon the statistical significance of the difference. It was also noted that the C4.5 tree used only 10 of the 38 inputs whereas the linear discriminant classifier used 17 inputs to make the classification decision.

In [117], Henley and Hand propose an adjusted Euclidean distance metric for a k-nearest neighbor credit scoring system (see section 2.1.8 of chapter 2). The experiments were carried out on a data set of 19186 customers which applied for mail-order credit. Each applicant was described by 16 categorical inputs. Applicants who defaulted for three consecutive months were classified as bad. All performance calculations were done on 5 randomizations of all available data thereby each time using 80% of the observations for training and the remaining 20% for testing. Comparisons were made with linear regression, logistic regression, projection-pursuit regression[3] and decision trees. It was concluded that the k-nearest neighbor method marginally outperformed the other methods.

Yobas et al. [264] compare the predictive performance of linear discriminant analysis, neural networks, genetic algorithms and C4.5 decision trees on a data set of 1001 credit card payers with 14 inputs. A bad applicant was defined as a customer who has missed one or more repayments. The performance was calculated using leave-one-out and 10-fold cross-validation. It was concluded that the linear discriminant analysis classifier yielded the best classification accuracy.

In [178], Piramuthu suggested the use of a filter input selection method to improve the classification accuracy of decision trees induced by C4.5. The experiments were, amongst other, conducted on the UCI Australian credit data set (653 Obs. and 14 inputs) using a 10-fold cross-validation setup. It was found that, for

[3]In projection-pursuit regression, one estimates a model consisting of a sum of smoothed functions of linear combinations of the inputs as follows:

$$y = \sum_{m=1}^{M} g_m(\omega_m^T \mathbf{x}). \tag{3.1}$$

The functions $g_m(\cdot)$ are called ridge functions. If M is chosen arbitrarily large, it can be shown that the projection-pursuit regression model becomes a universal approximator. A two-step estimation procedure is used to estimated $g_m(\cdot)$ and ω_m interchangeably. For more details, see [93, 112].

this data set, the proposed input selection method improved the predictive power of the decision tree and also decreased its size. The same data set was used in [179] where a comparison was made between neural networks and neurofuzzy systems for credit scoring. The data set was 10 times randomly split into a training set (490 Obs.) and a test set (163 Obs.). It was concluded that neural networks yielded a better classification accuracy than neurofuzzy systems but the latter have the additional benefit of generating comprehensible fuzzy *If Then* rules which allow one to explain the classification decision.

Desai et al. [62] contrasted the performance of multilayer perceptron neural networks, modular neural networks, linear discriminant analysis and logistic regression for three credit unions in the Southeastern United States for the period 1988 through 1991. After preprocessing, the data sets consisted of 505, 762 and 695 observations, respectively, each with 18 inputs. A bad applicant was defined as someone whose most recent loan was either charged off or someone who went bankrupt during the last 48 months. All other customers were defined as good provided that their most recent loan was between 48 months and 18 months old. The classification techniques were evaluated using ten randomizations of the data each time using 2/3 of the observations for training and the remaining 1/3 for testing. Furthermore, the authors also investigated the performance differences between customized credit scoring models, specifically tailored to the credit union under investigation, and generic credit scoring models which aim at developing one overall classification model for all three unions. It was found that customized neural networks performed very well at classifying the bad loans. Logistic regression and neural networks yielded the same performance when looking at the percentage of good and bad loans correctly classified. The performance of the generic models was not so good as the customized models especially for the bad loans. This research was further extended in [61] where genetic algorithms were also included in the study and a three-way classification was adopted categorizing customers as either good, poor or bad. A customer was classified as good if there were no payments that had been overdue for 31 days or more, poor if the payment had ever been overdue for 60 days or more, and bad if, at any time in the last 48 months, either the customer's most recent loan was charged off or the customer went bankrupt. Using the same experimental setup as in [62], and only focussing on customized credit scoring models, it was found that the linear discriminant analysis and logistic regression classifier compared favorably to the neural network and genetic algorithm classifier. Neural networks performed somewhat better at classifying the poor loans.

In [220], Steenackers and Goovaerts use logistic regression to build a scorecard for loan data obtained from a Belgian credit company. The loans dated from November 1984 till December 1986. A loan was considered to be bad after three reminders. The data contained 2940 observations with 19 inputs and was split into a training set of 2300 observations and a test set of 640 observations. Reject inference was done by first estimating the logistic regression classifier on the sample of accepted applicants and then using this model to score and label the rejected applicants. The final scorecard is then estimated on the entire population consisting of the accepted applicants with their observed outcomes and the rejected applicants with their predicted outcomes. No comparisons were made with other classification techniques.

Arminger et al. [7] compare the classification accuracy of logistic regression, decision trees and neural networks on a data set of 8163 consumer loans obtained from a major financial institution in Germany. All classification models are constructed using 6 inputs. The data set was split into a training set, validation set and test set whereby the validation set was used to avoid overfitting of the decision tree and neural network. It was concluded that the performance differences between the three techniques are small with the logistic regression classifier as the best technique. The authors also propose a way to combine the forecasts of the 3 techniques but this yielded no significant performance benefits.

West [259] investigates the use of five different types of neural networks for credit scoring: multilayer perceptron, mixture-of-experts, radial basis function, learning vector quantization and fuzzy adaptive resonance. The performance of these neural networks is contrasted with the performance of linear discriminant analysis, logistic regression, k-nearest-neighbor, kernel density estimation and decision trees. The experiments are carried out on the Statlog German credit data set (1000 Obs., 24 inputs) and the UCI Australian credit data set (690 Obs., 14 inputs) using a 10-fold cross-validation setup. The author also investigates the impact of misclassification costs. The best performance was obtained by the mixture-of-experts neural network, the radial basis function neural network and the logistic regression classifier.

Chang et al. [44] used Bayesian networks to graphically represent the conditional dependencies and independencies in a credit scoring data set obtained from a large U.K. bank. The data set consisted of 7104 observations with 34 inputs and was split into a training set of 5000 records and a test set of 2104 records. The AUC of the constructed networks was contrasted with the AUC of logistic regression using various input subsets. It was concluded that both techniques yielded approximately the same performance and should be used in combination.

Note that this literature overview is by no means exhaustive. Other methods that have been used for credit scoring are, e.g., genetic algorithms [24, 88], mathematical programming [89, 106, 140], Markov Chains [56, 95, 149], and (graphical) conditional independence models [108, 212, 219].

Most of the above studies are limited in terms of both the number of data sets considered and the number of techniques implemented. Furthermore, the use of the AUC as a performance criterion is also not widespread in the credit scoring literature. In the following section, we will conduct a large scale benchmarking study, similar to the one reported in section 2.4, on 8 credit scoring data sets.

3.4 Building Scorecards for Credit Scoring

3.4.1 Data Sets and Experimental Setup

Table 3.1 displays the characteristics of the data sets that will be used to evaluate the performance of the different classification techniques [13]. The Bene1 and Bene2 data sets were obtained from two major financial institutions in the Benelux (Belgium, The Netherlands and Luxembourg). The UK1 and UK2 data sets are from the same financial institution located in the U.K. but with other class distributions. Also the UK3 and UK4 data sets are from the same U.K. financial institution but with different class proportions. For the Bene1, Bene2, UK1, UK2, UK3 and UK4 data sets, a bad customer was defined as someone who has missed three consecutive months of payments. This is the definition that is commonly adopted in the credit scoring industry [234]. The German credit and Australian credit data sets are publicly available at the UCI repository (`http://kdd.ics.uci.edu/`). Note that we use a somewhat different setup from the one used in section 2.4. The reason is that most data sets are quite large and the variation that would be obtained by resampling would be very small. Hence, we use a training set/test set setup whereby each data set is split into 2/3 training set and 1/3 test set. The classifiers are trained on the training set and evaluated on the test set.

	Inputs	Data set size	Training set size	Test set size	Goods/Bads
Bene1	33	3123	2082	1041	66.7/33.3
Bene2	33	7190	4793	2397	70/30
UK1	16	9360	6240	3120	75/25
UK2	16	11700	7800	3900	80/20
UK3	19	3960	2640	1320	90/10
UK4	19	1980	1320	660	80/20
Germ	20	1000	666	334	70/30
Austr	14	690	460	230	55.5/44.5

Table 3.1: Characteristics of credit scoring data sets.

The topic of choosing the appropriate class proportions for classifier learning has received much attention in the literature [258]. In this benchmarking study, we use a variety of class distributions ranging from 55.5/44.5 for the Australian credit data set to 90/10 for the UK3 data set.

Since the C4.5 and C4.5rules classifiers may yield better results with discretized data, we will conduct experiments using both the (original) continuous data and the discretized data. The discretization process will be carried out using the discretization algorithm of Fayyad and Irani with the default options [83]. This algorithm uses an information entropy minimization heuristic to discretize the range of continuous-valued attributes into multiple intervals. The discretized data will also be used to train the TAN classifiers.

We encoded a nominal variable having r values by using $r-1$ (binary) dummy variables. The continuous inputs are normalized to zero mean and unit variance according to Equation 2.55. The following techniques discussed in chapter 2 will be evaluated: linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), logistic regression (LOG), linear programming (LP), least squares support vector machines (LS-SVMs), standard support vector machines (SVMs), neural networks (NN), naive Bayes (NB), tree augmented naive Bayes (TAN), C4.5, C4.5rules, K-nearest neighbor with K=10 (KNN10), and K-nearest neighbor with K=100 (KNN100). The LDA, QDA, LOG and LP classifiers require no parameter tuning. For the LS-SVM and SVM classifiers, we used both linear and RBF kernels and adopted the grid search mechanism reported in section 2.4.2 to tune the hyperparameters γ and σ. The SVM and LS-SVM analyses were carried out using the Matlab^{TM} toolbox of Gavin Cawley [4] and the LS-SVMlab Toolbox [5], respectively. The NN classifiers were trained using the Bayesian evidence framework of David MacKay with the ARD extension [154, 155]. We used only 1 hidden layer influenced by theoretical works and varied the number of hidden neurons from 1 to 10. We then chose the network with the best training set accuracy for evaluation on the test set. All NN analyses were conducted using the Netlab toolbox[6]. For the naive Bayes classifier, we use the kernel approximation for continuous attributes. The naive Bayes and TAN analyses were done using the Bayes Net Toolbox of Kevin Murphy[7]. Again, we set the confidence level for the pruning strategy of C4.5 and C4.5rules to 0.25 which is the default value.

3.4.2 Setting the cut-off

When implementing a scorecard, one needs to choose a cut-off to map the scores $P(y = 1|\mathbf{x})$ to class labels $y = 1$ or $y = 0$. Probably the most simple way is to set the cut-off in the middle of the score spectrum. E.g., for the logistic regression classifier, one classifies an observation with characteristics $\mathbf{x}$ into class 1 if $P(y = 1|\mathbf{x}) > 0.5$ and into class 0 otherwise. A major drawback of this naive method is that it fails to properly take into account the prior class distributions and the misclassification costs. E.g., when confronted with a very skew class distribution, which is often the case in credit scoring and fraud detection, it can easily be that $P(y = 1|\mathbf{x}) >> 0.5$ for all values of $\mathbf{x}$ which would mean that all customers would be classified as good ($y = 1$) and none as bad ($y = 0$) [257]. Although a high overall classification accuracy may be obtained in this way, the

[4]`http://theoval.sys.uea.ac.uk/~gcc/svm/`

[5]`http://www.esat.kuleuven.ac.be/sista/lssvmlab/`

[6]`http://www.ncrg.aston.ac.uk/netlab/`

[7]`http://www.ai.mit.edu/ murphyk/Software/BNT/bnt.html`

most risky and costly class remains undetected.

Ideally, when setting the cut-off, one should take into account the misclassification costs i.e. the cost of misclassifying a bad as good versus the cost of misclassifying a good as bad. If one would know these numbers, an observation could be classified into class 1 if

$$P(y=1|\mathbf{x})C_{11}+P(y=0|\mathbf{x})C_{10} \leq P(y=1|\mathbf{x})C_{01}+P(y=0|\mathbf{x})C_{00}, \tag{3.2}$$

whereby C_{ij} represents the misclassification cost of classifying an object with actual class j as class i. When assuming that correct classifications incur no cost ($C_{11}=C_{00}=0$) this boils down to assigning a customer to class 1 if

$$\begin{aligned} P(y=1|\mathbf{x}) &\geq \tfrac{C_{10}}{C_{10}+C_{01}} \\ &\geq \tfrac{1}{1+\frac{C_{01}}{C_{10}}}. \end{aligned} \tag{3.3}$$

Note that one only needs to know the ratio of both costs $\frac{C_{01}}{C_{10}}$ in order to make a classification decision. When this ratio is large, the cut-off will be very different from 0.5.

Although choosing a cut-off based on the ratio of the misclassification costs is attractive from an economic perspective, it has to be noted that specifying the exact values of the costs C_{ij} is usually very difficult since they are typically dependent upon the characteristics of the loan, e.g. amount, purpose, interest rate, and thus vary from customer to customer. Furthermore, due to the changing economic climate, the costs may also vary over time making their correct quantification less obvious. Another issue which complicates the setting of the cut-off is the sample which was used to train the classifier. Equation 3.2 implicitly assumes that the sample which was used to train the classifier and estimate $P(y=1|\mathbf{x})$ was taken in such a way such that the a priori class proportions $P(y=1)$ and $P(y=0)$ are unbiased estimates of the true underlying proportions [257]. If the sample is not representative for the true underlying population due to e.g. an oversampling of the bads or undersampling of the goods, one may have to apply score adjustments before using the scorecard in practice [198]. Hence, the issues of misclassification costs and sample construction often necessitate one to rely on heuristically based cut-off setting schemes.

A first alternative cut-off setting scheme is to set the cut-off such that the number of predicted bads (goods) equals the actual number of bads (goods) in the sample. Another method which is also often used in practice is based upon the marginal good-bad rate. This scheme sets the cut-off at the point where the marginal rate of goods to bads if it is dropped anymore goes below $m:1$ whereby common values for m are 3 and 5 [20, 235]. This is illustrated in the following example.

Example 3.1
Consider the run-book of Table 3.2 depicting the effect of cut-off changes on the popu-

lation. When assuming a marginal good-bad rate of 5:1 (3:1), the cut-off is set at 0.6 (0.5), respectively.

Cut-off	**Cum. goods above score**	**Cum. bads above score**	**Marginal good-bad rate**
0.9	1400	50	-
0.8	2000	100	12:1
0.7	2700	170	10:1
0.6	2950	220	5:1
0.5	3130	280	3:1
0.4	3170	300	2:1

Table 3.2: The marginal good-bad rate.

A major criticism on the use of the marginal good-bad rate method is that it is heavily dependent upon the granularity of the cut-off change. In the next subsection, we will present the results using three types of cut-off setting schemes: using a cut-off of 0.5; using a cut-off such that the number of predicted bads (goods) equals the actual number of bads (goods) and using cut-offs based on marginal good bad rates of 5:1 and 3:1.

3.4.3 Results

Tables 3.3, 3.4, 3.5 and 3.6 report the test set PCC, sensitivity and specificity of all classifiers on the eight credit scoring data sets using the four cut-off setting schemes discussed in the previous subsection. We have used a special notational convention whereby the best test set PCC performance per data set is underlined and denoted in bold face, performances that are not significantly different at the 5% level from the top performance with respect to a one-tailed McNemar test are tabulated in bold face, and statistically significant underperformances at the 1% level are emphasized in italics. Performances significantly different at the 5% level but not at the 1% level are reported in normal script. We also compute a ranking of the different algorithms on each data set assigning rank 1 to the algorithm yielding the best PCC and rank 17 to the algorithm giving the worst PCC. These ranks are then averaged and compared using a Wilcoxon signed rank test of equality of medians (see last column of Tables 3.3, 3.4, 3.5 and 3.6).

It can be seen from Table 3.3 that, for the first cut- off scheme, the RBF LS-SVM classifier yielded the best average ranking (AR) for the PCC measure. The LOG, LP, Lin LS-SVM, NN, C4.5 dis and KNN100 classifiers have statistically the same AR for the PCC as the RBF LS-SVM classifier at the 5% significance level. The PCC average rankings of the QDA, NB, TAN, and C4.5rules classifiers are statistically inferior to the PCC AR of the RBF LS-SVM classifier at the 1% level. It can also be observed that both the C4.5 and C4.5rules classifiers achieve better average PCC rankings on the discretized data than on the continuous data.

Remember that the performance measures reported in Table 3.3 were calculated assuming a cut-off value of 0.5 on the classifier's output. This may however not be the most appropriate threshold to use for the more skewed U.K. data sets. For these data sets, some classifiers have a tendency to always predict the majority class (good customer) yielding hereby 100% sensitivity and 0% specificity. Especially the KNN100 classifier tends to predict the majority class for many of the data sets considered. Hence, it might also be interesting to look at the results of Table 3.4 assuming a cut-off such that the number of predicted bads (goods) equals the actual number of bads (goods). Table 3.4 indicates that the NN classifier achieved the best average rank with those of the LOG, RBF LS-SVM, Lin LS-SVM, Lin SVM, NB, and TAN not being statistically different from it at the 5% level. The KNN100 classifier now achieves better specificity values on the U.K. data sets. However, the C4.5(dis) classifiers yield very bad sensitivity values on the U.K. data sets. This can be explained by the fact that, due to the skewness of these data sets, the C4.5(dis) trees are unable to grow a good tree and tend to predict the majority class.

Table 3.5 and 3.6 present the results with cut-offs based on a marginal good-bad rate of about 5:1 and 3:1, respectively [235, 20]. The cut-offs were calculated by lowering the cut-off from 1 to 0 in steps of 0.05 until the marginal good-bad rate reached the desired level. Using the 5:1 scheme, it was found that the Lin SVM classifier gave the best PCC AR although it achieved very bad specificity values on the U.K. data sets. The average rank of the LP, RBF LS-SVM, Lin LS-SVM, NN, and C4.5 dis classifiers are not statistically different from that of the Lin SVM classifier at the 5% level. Note that also the LP and C4.5 dis classifiers yielded very bad specificity values for the U.K. data sets. For the 3:1 scheme, the RBF LS-SVM classifier yielded the best average rank with those of the LDA, LP, Lin LS-SVM, RBF SVM, Lin SVM, and C4.5 dis classifiers not being statistically different from it at the 5% level. Again remark however that the LP, Lin SVM and C4.5 dis classifiers yielded very bad specificity or sensitivity values for the U.K. data sets. Also note that the 3:1 cut-off scheme, when compared to the 5:1 scheme, generally resulted in lower cut-offs and thus increased sensitivity and decreased specificity values.

Table 3.7 reports the AUCs of all classifiers on the eight credit scoring data sets. This table has the same setup as the PCC Tables. It clearly indicates that the RBF LS-SVM classifier was two times ranked first, the NN classifier four times and the Lin SVM and TAN classifiers each one time. It can be observed that the best average rank was attributed to the NN classifier. The AR of the LDA, LOG, RBF LS-SVM, Lin LS-SVM, and TAN classifiers are not statistically inferior at the 5% significance level. The AR of the QDA, LP, C4.5, C4.5rules, C4.5dis, and KNN10 classifiers are statistically worse than the AR of the NN classifier at the

1% level. Note that for the Bene2, UK1 and UK2 data sets, the NN classifier always performed significantly better than the LOG and LDA classifiers in terms of AUC. Although the difference may be small in absolute terms, it has to be noted that in a credit scoring context, an increase in the discrimination ability of even a fraction of a percent may translate into significant future savings[117].

3.4.4 Discussion

When looking at the results depicted in Tables 3.3, 3.4, 3.5 and 3.6, it becomes clear that the ranking of the classifiers highly depends upon the chosen cut-off strategy. Hence, these tables should be interpreted in combination with the AUC values reported in Table 3.7. Some general trends can be observed. Many classification techniques yield performances which are quite competitive with each other. No single classification technique emanates as the best for all criteria on all data sets. This phenomenon is in the literature often described as the *flat maximum effect* and was also found by Thomas as the following quote illustrates:

> *"it is surprising that the recurring theme is how little difference there is in the classification accuracy of the different approaches [107]".*

One possible explanation for this phenomenon is that it is usually very difficult to achieve a good separation between good and bad customers. Credit scoring data sets are typically very noisy. Two customers with the same characteristics can easily belong to different classes [104]. Another issue is that scorecard developers are well aware of the discriminating characteristics of both classes, and there is little extra knowledge to be modeled by the sophisticated classification techniques [104].

From the performance tables it can be concluded that especially the non-linear RBF LS-SVM and NN classifiers consistently yield a very good performance. However, the more simple, linear classification techniques such as LDA and LOG also have a very good performance which is in the majority of the cases not statistically different from that of the RBF LS-SVM and NN classifiers. This finding clearly indicates that most credit scoring data sets are only weakly non-linear. Only a handful of classifiers (e.g. QDA, NB and C4.5rules) yielded weak performances in terms of both PCC and AUC whereas others (e.g. LP and C4.5 dis) gave a good PCC performance but a rather bad AUC.

It needs also be noted that small (statistically significant) differences in classification performance (PCC or AUC), even a fraction of a percent may, in a credit scoring context, translate into significant future savings. This is illustrated by the following quotes of Henley and Hand [117] and Kelly [135]:

> *"Although the differences are small, they may be large enough to have*

Technique	Bene1			Bene2			Germ			Austr			UK1			UK2			UK3			UK4			AR
	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC
LDA	**72.2**	82.3	53.2	73.7	87.7	40.9	**74.6**	90.0	41.0	**88.3**	87.4	89.3	74.4	98.3	3.56	80.2	98.0	6.83	*88.6*	98.6	3.60	**82.0**	96.1	21.0	6.94
QDA	*56.9*	40.9	87.2	*41.2*	17.3	96.9	**71.0**	79.5	52.4	*83.0*	92.9	70.9	*71.5*	89.4	18.4	*77.8*	91.4	22.1	*84.6*	93.0	13.7	*77.7*	89.9	25.0	*15.8*
LOG	**72.0**	82.3	52.6	**74.4**	87.8	43.1	**74.6**	89.5	41.9	87.4	91.3	82.5	*74.4*	98.4	2.93	80.3	98.5	5.12	**89.2**	99.5	1.44	**82.1**	96.3	21.0	**6.06**
LP	71.2	79.8	54.9	**74.2**	88.4	40.9	**71.9**	92.6	26.7	**88.3**	86.6	90.3	**74.8**	100	0.00	**80.5**	100	0.00	**89.5**	100	0.00	**81.2**	100	0.00	**6.50**
RBF LS-SVM	**73.1**	83.9	52.6	**74.3**	88.4	41.3	**74.3**	96.5	25.7	**89.1**	89.8	88.3	**74.8**	100	0.00	**80.7**	99.7	2.23	**89.0**	99.1	3.60	**82.1**	97.4	16.1	**3.56**
Lin LS-SVM	71.4	84.3	46.8	73.7	89.5	36.9	**73.7**	91.3	35.2	**88.3**	87.4	89.3	**74.6**	99.2	1.65	80.4	99.5	1.58	**89.5**	100	0.00	**81.8**	97.9	12.1	**6.75**
RBF SVM	71.9	82.1	52.4	73.9	89.2	38.2	**74.0**	92.6	33.3	87.0	89.8	83.5	**74.8**	99.9	0.25	80.2	99.1	2.37	**89.2**	99.3	2.88	**81.2**	98.9	4.84	8.19
Lin SVM	71.2	82.1	50.4	73.9	88.4	40.2	71.0	95.6	17.1	**88.3**	86.6	90.3	**74.8**	100	0.00	**80.5**	100	0.00	**89.5**	100	0.00	**81.2**	100	0.00	6.94
NN	**72.4**	81.8	54.6	**75.1**	86.7	48.1	**73.7**	85.2	48.6	**88.3**	92.1	83.5	**75.0**	99.2	3.05	**80.6**	99.9	0.92	**89.4**	99.7	2.16	**80.9**	93.3	27.4	**5.19**
NB	*65.8*	55.1	86.1	*59.0*	51.1	77.5	**72.2**	87.8	38.1	*82.2*	100	60.2	*70.8*	87.2	22.1	*77.8*	90.6	25.4	*88.1*	97.1	11.5	*77.9*	88.4	32.3	*15.1*
TAN	*69.5*	78.2	53.2	73.4	82.7	51.7	**72.5**	87.3	40.0	87.4	92.9	80.6	*72.9*	93.9	10.6	*77.4*	90.6	22.9	*87.1*	96.2	10.1	*76.8*	87.9	29.0	*12.9*
C4.5	*68.9*	77.4	52.6	*69.8*	81.3	43.0	**72.2**	88.2	37.1	*84.3*	93.7	72.8	*71.1*	89.3	16.8	*79.3*	95.0	14.3	**89.5**	100	0.00	**81.2**	100	0.00	12.1
C4.5rules	**71.5**	80.4	54.6	*71.4*	76.8	58.8	**71.0**	91.3	26.7	*85.2*	90.6	78.6	*67.1*	77.4	36.6	80.4	99.9	0.13	*88.5*	98.6	2.16	**81.7**	97.6	12.9	*11.3*
C4.5 dis	*69.3*	78.6	51.5	73.1	85.3	44.6	**74.6**	87.3	46.7	**89.1**	94.5	82.5	**74.8**	100	0.00	**80.5**	100	0.00	**89.5**	100	0.00	**81.2**	100	0.00	**6.69**
C4.5rules dis	70.2	80.6	50.4	**73.9**	84.8	48.5	**74.3**	89.5	41.0	**90.4**	94.5	85.4	*72.7*	94.6	7.76	*79.8*	98.1	4.47	88.6	98.2	6.47	**80.8**	97.6	8.06	9.19
KNN10	*66.7*	83.9	34.0	*70.8*	92.2	21.0	**70.7**	94.8	18.1	86.5	93.7	77.7	*73.2*	96.1	5.22	*79.8*	98.1	4.20	**89.5**	100	0.00	**81.2**	97.9	8.87	11.9
KNN100	70.3	83.6	45.1	*72.0*	96.2	15.3	*68.6*	100	0.00	**88.7**	95.3	80.6	**74.8**	100	0.13	**80.5**	100	0.00	**89.5**	100	0.00	**81.2**	100	0.00	**7.94**

Table 3.3: Test set classification accuracy on credit scoring data sets assuming a cut-off of 0.5.

Technique	Bene1			Bene2			Germ			Austr			UK1			UK2			UK3			UK4			AR
	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC
LDA	**71.4**	78.2	58.5	**73.0**	80.7	54.9	**73.1**	80.3	57.1	**87.8**	89.0	86.4	**67.8**	78.4	36.0	*74.6*	84.2	34.8	**85.3**	91.8	30.2	**77.6**	86.2	40.3	6.00
QDA	*69.6*	76.8	56.0	*70.1*	78.7	50.2	**70.7**	78.6	53.3	*83.5*	85.0	81.6	66.6	77.7	33.7	74.8	84.3	35.3	*82.9*	90.4	18.7	75.5	84.9	34.7	*11.1*
LOG	**71.8**	78.4	59.1	**73.6**	81.2	56.1	**73.1**	80.3	57.1	87.0	88.2	85.4	**68.0**	78.6	36.5	74.9	84.4	35.7	**85.5**	91.9	30.9	**78.2**	86.6	41.9	**4.50**
LP	**70.6**	77.6	57.4	**73.0**	80.8	55.1	**70.7**	78.6	53.3	**87.8**	89.0	86.4	*63.2*	75.4	27.0	*67.8*	80.0	17.5	**83.6**	90.9	22.3	**75.5**	84.9	34.7	*11.1*
RBF LS-SVM	**72.1**	78.7	59.6	**72.8**	80.6	54.7	**73.1**	80.3	57.1	**88.7**	89.8	87.4	**67.4**	78.2	35.4	75.0	84.5	36.0	83.8	90.9	23.0	**77.9**	86.4	41.1	**5.38**
Lin LS-SVM	**71.2**	78.0	58.2	**73**	80.7	54.9	**73.1**	80.3	57.1	**87.8**	89.0	86.4	**68.1**	78.7	36.8	74.7	84.3	35.2	**84.7**	91.4	27.3	**78.2**	86.6	41.9	**5.63**
RBF SVM	**71.4**	78.2	58.5	72.5	80.3	54.1	**73.1**	80.3	57.1	**87.8**	89.0	86.4	67.1	78.0	34.7	*72.2*	82.7	28.6	**83.9**	91.0	23.7	**76.4**	85.4	37.1	8.06
Lin SVM	**71.0**	77.9	57.9	**73.1**	80.8	55.2	**71.3**	79.0	54.3	**89.6**	90.6	88.3	*63.6*	75.7	27.7	*73.3*	83.4	31.7	**84.1**	91.1	24.5	74.5	84.3	32.3	**8.44**
NN	**71.4**	78.2	58.5	**73.8**	81.3	56.3	**74.3**	81.2	59.0	**87.8**	89.0	86.4	**68.1**	78.7	36.6	**76.1**	85.2	38.8	**84.8**	91.5	28.1	**77.9**	86.4	41.1	**3.69**
NB	**72.5**	79.0	60.2	*69.1*	77.9	48.5	**73.1**	80.3	57.1	**89.6**	90.6	88.3	**68.1**	78.7	36.8	**75.3**	84.6	36.7	**84.2**	91.2	25.2	**76.7**	85.6	37.9	**4.75**
TAN	*69.3*	76.2	56.0	**73.2**	80.9	55.4	**72.5**	79.9	56.2	**88.7**	89.8	87.4	**68.8**	79.1	38.4	**75.6**	84.9	37.6	**83.8**	90.9	23.0	**75.5**	84.9	34.7	**6.31**
C4.5	*68.7*	75.8	55.2	*69.2*	75.4	54.8	68.9	70.3	65.7	**88.7**	88.2	89.3	*63.2*	70.5	41.6	*72.6*	79.0	46.5	*10.5*	0.00	100	*18.8*	0.00	100	*13.3*
C4.5rules	**71.0**	76.7	60.2	*71.4*	76.8	58.8	*55.4*	53.3	60.0	*46.5*	9.45	92.2	*65.7*	74.3	40.3	*73.5*	80.6	43.9	*41.7*	37.8	75.5	*73.0*	78.9	47.6	*13.0*
C4.5 dis	*69.1*	76.4	55.2	*71.4*	79.3	52.9	**72.2**	78.6	58.1	**88.3**	86.6	90.3	*25.2*	0.00	100	*19.5*	0.00	100	*10.5*	0.00	100	*18.8*	0.00	100	*13.6*
C4.5rules dis	*66.0*	59.8	77.7	*60.0*	53.2	75.8	*63.5*	69.4	50.5	**89.6**	89.0	90.3	*66.2*	73.5	44.5	*69.4*	73.1	53.7	*82.6*	89.0	28.1	*73.9*	83.0	34.7	12.6
KNN10	*65.4*	63.2	69.6	*68.1*	75.0	52.0	68.6	73.8	57.1	*83.9*	81.9	86.4	*56.8*	57.9	53.6	*66.9*	70.8	50.9	*75.1*	79.9	33.8	*67.9*	71.6	51.6	*15.5*
KNN100	**71.3**	77.3	59.9	*68.8*	75.8	52.3	**71.9**	77.3	60.0	**88.7**	89.8	87.4	*66.0*	75.7	37.0	*72.8*	81.8	35.9	83.2	89.8	26.6	**75.5**	84.3	37.1	10.0

Table 3.4: Test set classification accuracy on credit scoring data sets assuming equal sample proportions.

Technique	Bene1			Bene2			Germ			Austr			UK1			UK2			UK3			UK4			AR
	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC
LDA	*62.4*	48.4	89.1	*62.2*	51.4	87.3	*58.7*	42.8	93.3	*86.5*	83.5	90.3	*50.7*	41.3	78.6	*78.5*	94.2	14.1	*85.3*	91.8	30.2	*68.0*	68.8	64.5	*9.44*
QDA	*56.3*	39.7	87.7	*40.9*	16.9	96.9	**71.0**	79.5	52.4	*86.1*	90.6	80.6	*49.2*	38.1	82.3	*64.3*	61.8	74.5	*83.9*	91.8	16.5	*77.1*	88.1	29.8	*10.8*
LOG	*58.3*	40.9	91.4	*62.2*	51.5	87.3	*65.0*	55.9	84.8	*84.8*	80.3	90.3	*50.0*	40.1	79.4	*66.3*	65.3	70.0	**88.9**	98.6	5.76	*74.1*	77.8	58.1	*9.25*
LP	*59.1*	42.1	91.4	*58.3*	43.8	92.1	*64.7*	56.3	82.9	88.3	86.6	90.3	**74.8**	100	0.00	**80.5**	100	0.00	**89.5**	100	0.00	**81.2**	100	0.00	**5.06**
RBF LS-SVM	**67.1**	58.1	84.4	67.4	62.8	78.3	**75.1**	87.3	48.6	*47.8*	6.30	99.0	*73.1*	93.6	12.3	*74.1*	81.6	43.2	*87.0*	95.9	12.2	*76.1*	82.1	50.0	**5.88**
Lin LS-SVM	65.3	54.7	85.5	*65.8*	60.0	79.4	*58.7*	42.8	93.3	*87.4*	85.0	90.3	*43.9*	29.5	86.6	*75.3*	85.2	34.4	*86.7*	94.6	20.1	*76.4*	82.8	48.4	**8.19**
RBF SVM	*62.0*	47.5	89.4	*62.2*	51.4	87.3	*63.5*	54.6	82.9	*85.2*	81.9	89.3	*26.1*	1.41	99.2	*74.6*	88.2	18.8	*86.9*	95.4	14.4	*77.1*	87.1	33.9	*8.94*
Lin SVM	*61.2*	46.9	88.3	*62.9*	53.2	85.5	*66.8*	58.5	84.8	88.3	86.6	90.3	**74.8**	100	0.00	**80.5**	100	0.00	**89.5**	100	0.00	**81.2**	100	0.00	**3.94**
NN	**67.1**	58.1	84.1	*64.1*	53.8	88.2	**70.4**	68.6	74.3	*81.3*	74.8	89.3	*49.3*	36.8	86.3	*61.1*	55.0	86.2	*80.0*	84.5	41.7	*61.5*	59.3	71.0	**10.3**
NB	*57.7*	40.2	91.1	*57.0*	47.5	79.1	*62.6*	52.4	84.8	87.4	93.7	79.6	*52.7*	43.9	78.8	*66.9*	65.8	71.4	*84.1*	90.9	25.9	*76.7*	85.6	37.9	*10.4*
TAN	*63.7*	50.6	88.6	*64.0*	54.6	85.8	*63.2*	51.5	88.6	*81.3*	74.0	90.3	*45.4*	29.2	93.3	*61.2*	55.5	85.0	*86.7*	95.3	13.7	*75.3*	83.8	38.7	9.94
C4.5	*59.7*	46.2	85.2	*31.1*	1.91	99.3	68.9	68.1	70.5	88.7	88.2	89.3	*61.4*	66.0	47.8	*71.3*	75.5	53.7	**89.4**	99.9	0.00	**81.1**	99.8	0.00	7.06
C4.5rules	*50.5*	28.6	92.2	**69.7**	70.5	67.7	*45.5*	27.5	84.8	*46.5*	9.45	92.2	*53.3*	45.5	76.3	*67.9*	67.7	68.9	*88.5*	98.6	2.16	*60.0*	58.2	67.7	11.4
C4.5 dis	*53.2*	33.0	91.6	*61.9*	54.1	80.1	*64.1*	57.6	78.1	89.1	94.5	82.5	**74.8**	100	0.00	**80.5**	100	0.00	**89.5**	100	0.72	**81.1**	99.8	0.00	**5.81**
C4.5rules dis	*52.7*	31.1	93.9	*50.5*	34.3	88.3	**74.3**	89.5	41.0	**91.7**	94.5	88.3	*72.7*	94.6	7.76	*57.7*	50.6	87.3	*84.9*	92.4	21.6	*73.5*	81.0	41.1	10.0
KNN10	*62.6*	51.6	83.6	*53.5*	39.7	85.8	*41.3*	17.0	94.3	*81.3*	74.0	90.3	*66.2*	77.5	32.4	*55.2*	50.8	73.1	*85.1*	93.6	12.9	*67.9*	71.6	51.6	*12.3*
KNN100	*56.2*	37.2	92.2	*57.1*	45.2	85.0	*62.0*	52.0	83.8	*79.6*	69.3	92.2	*44.5*	29.3	89.6	*58.3*	54.0	76.3	*83.2*	89.8	26.6	*63.5*	62.5	67.7	*14.4*

Table 3.5: Test set classification accuracy on credit scoring data sets assuming a marginal good-bad rate around 5:1.

Technique	Bene1			Bene2			Germ			Austr			UK1			UK2			UK3			UK4			AR
	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC	Sens	Spec	PCC
LDA	*66.1*	56.6	84.1	*65.8*	60.6	78.0	*65.3*	57.2	82.9	88.7	87.4	90.3	*72.3*	92.4	12.7	*72.4*	77.6	50.9	88.6	98.1	7.91	**81.4**	94.4	25.0	**7.44**
QDA	*56.3*	39.7	87.7	*40.9*	16.9	96.9	68.6	73.4	58.1	*85.7*	92.1	77.7	*70.4*	86.2	23.2	*76.7*	88.3	28.8	*83.9*	91.8	16.5	*77.4*	88.6	29.0	12.1
LOG	*68.4*	61.7	81.1	*66.2*	59.8	81.2	**71.6**	72.5	69.5	87.8	86.6	89.3	*60.0*	59.3	62.2	*71.9*	77.2	50.5	*87.7*	95.7	20.1	**79.2**	88.6	38.7	*8.88*
LP	*62.7*	50.3	86.4	*67.5*	63.6	76.5	**73.4**	85.2	47.6	88.3	86.6	90.3	**74.8**	100	0.00	**80.5**	100	0.00	*10.5*	0.00	100	**81.2**	100	0.00	**6.19**
RBF LS-SVM	**72.3**	72.4	72.1	*67.4*	62.8	78.3	**75.1**	87.3	48.6	89.1	89.8	88.3	*73.1*	93.6	12.3	*74.1*	81.6	43.2	*88.3*	98.0	5.76	**81.1**	93.1	29.0	**4.56**
Lin LS-SVM	*65.3*	54.7	85.5	*65.8*	60.0	79.4	*67.1*	64.6	72.4	88.3	87.4	89.3	*66.4*	74.0	43.6	*75.3*	85.2	34.4	**89.4**	99.7	2.16	*76.4*	82.8	48.4	**8.63**
RBF SVM	*69.6*	65.7	77.2	*66.7*	61.9	78.0	**73.7**	79.5	61.0	87.8	88.2	87.4	*73.8*	96.7	6.11	*74.6*	88.2	18.8	**89.2**	99.3	2.88	**81.5**	95.5	21.0	**4.94**
Lin SVM	*64.6*	54.5	83.8	*67.4*	63.2	77.2	69.5	71.2	65.7	88.3	86.6	90.3	**74.8**	100	0.00	**80.5**	100	0.00	*10.5*	0.00	100	**81.2**	100	0.00	**6.56**
NN	*63.3*	51.0	86.6	*66.8*	60.1	82.5	*65.0*	56.3	83.8	*87.8*	89.8	85.4	*45.6*	29.6	93.0	*66.3*	64.1	75.2	**88.9**	98.8	5.04	78.9	88.4	37.9	11.3
NB	*64.5*	52.3	87.5	*57.0*	47.5	79.1	**73.1**	84.3	48.6	*85.2*	98.4	68.9	*68.8*	80.2	34.9	*66.9*	65.8	71.4	*88.0*	96.7	14.4	*66.2*	70.7	46.8	*11.8*
TAN	*66.3*	56.3	85.2	**69.8**	66.7	77.1	**71.3**	69.4	75.2	88.3	88.2	88.3	*52.6*	42.4	82.7	*64.6*	61.7	76.3	*86.7*	95.3	13.7	*76.8*	87.9	29.0	*9.56*
C4.5	*63.3*	54.7	79.7	*65.7*	63.1	71.9	68.9	68.1	70.5	88.7	88.2	89.3	*71.1*	89.3	16.8	*71.3*	75.5	53.7	*10.5*	0.00	100	**81.1**	99.8	0.00	*9.94*
C4.5rules	**71.0**	76.7	60.2	*59.0*	48.6	83.3	*49.4*	35.8	79.0	*46.5*	9.45	92.2	*59.8*	59.6	60.4	*73.5*	80.6	43.9	*88.5*	98.6	2.16	**81.7**	97.6	12.9	10.1
C4.5 dis	69.6	68.0	72.7	*65.5*	63.2	70.9	**71.0**	76.0	60.0	89.1	94.5	82.5	**74.8**	100	0.00	**80.5**	100	0.00	*10.5*	0.00	100	**81.1**	99.8	0.00	**6.56**
C4.5rules dis	*62.2*	52.3	80.8	*55.6*	44.5	81.6	*63.5*	69.4	50.5	**91.7**	94.5	88.3	*57.1*	53.0	69.2	*61.2*	57.0	78.8	88.6	98.2	6.47	*73.0*	80.2	41.9	12.8
KNN10	*62.6*	51.6	83.6	*63.3*	60.0	71.1	68.6	73.8	57.1	87.4	91.3	82.5	*56.8*	57.9	53.6	*73.5*	84.2	29.3	**89.5**	99.8	1.44	*76.1*	86.4	31.5	11.2
KNN100	*64.1*	54.1	83.0	*62.0*	57.0	73.7	70.4	73.4	63.8	*83.0*	78.0	89.3	*52.7*	45.5	73.8	*79.2*	95.4	12.4	**89.4**	99.7	1.44	*73.2*	81.0	39.5	10.6

Table 3.6: Test set classification accuracy on credit scoring data sets assuming a marginal good-bad rate around 3:1.

Technique	Bene1	Bene2	Germ	Austr	UK1	UK2	UK3	UK4	AR
LDA	**77.1**	*77.1*	**78.4**	**92.8**	*64.1*	*73.6*	**74.4**	**72.3**	**5.38**
QDA	*73.4*	*72.4*	*71.8*	**91.5**	*63.3*	*72.1*	*68.1*	*68.3*	*10.8*
LOG	**77.0**	*78.0*	**77.7**	**93.2**	*63.9*	*73.0*	**74.6**	**72.7**	**4.38**
LP	*76.1*	*77.5*	**76.3**	**92.6**	*56.4*	*62.3*	*62.0*	*62.2*	*11.9*
RBF LS-SVM	**77.6**	77.8	**77.4**	**93.2**	65.0	74.7	**72.9**	**73.1**	**3.38**
Lin LS-SVM	**76.9**	*77.1*	**78.4**	**92.9**	*64.4*	*73.7*	**73.8**	**72.5**	**5.50**
RBF SVM	*76.7*	*77.1*	**77.2**	**92.6**	*59.3*	*65.4*	*67.3*	*68.4*	9.13
Lin SVM	*75.9*	*77.5*	**76.6**	**93.6**	*56.4*	*63.9*	*62.9*	*62.3*	10.1
NN	**76.9**	**79.1**	**78.7**	**91.7**	**66.4**	**75.8**	**74.6**	**72.9**	**3.25**
NB	**76.5**	*70.6*	**77.2**	**93.1**	**65.8**	*73.7*	*66.9*	67.9	7.88
TAN	75.5	**78.2**	**78.3**	**93.4**	**66.8**	74.5	*64.0*	*66.6*	**5.63**
C4.5	*72.2*	*71.1*	**74.7**	91.6	*56.1*	*65.7*	*50.0*	*49.9*	*14.7*
C4.5rules	*71.6*	*74.2*	*62.0*	*85.3*	*61.7*	*70.4*	*60.3*	68.4	*13.0*
C4.5 dis	*73.0*	*73.2*	**74.6**	**93.1**	*50.0*	*50.0*	*50.4*	*49.9*	*13.7*
C4.5rules dis	*73.0*	*71.5*	*64.4*	**93.1**	65.2	*71.5*	*66.7*	*64.9*	10.8
KNN10	*71.7*	*69.6*	*70.2*	91.4	*58.9*	*65.4*	*63.0*	67.0	*14.1*
KNN100	*74.9*	*71.5*	**76.1**	**93.0**	*62.8*	*69.9*	*70.0*	**70.4**	9.5

Table 3.7: Test set AUC on credit scoring data sets.

commercial implications [117]".

"Credit is an industry where improvements, even when small, can represent vast profits if such improvement can be sustained [135]".

We would also like to point out that it is our firm belief that the best way to augment the performance of a scorecard is not by merely looking for the best classification technique but also thoroughly investigating which predictors or inputs should be considered. In other words, the search for more powerful and discriminating inputs can yield substantial performance benefits. Financial institutions should try to collect additional information describing all kinds of characteristics of their loan applicants. It is in this context that many countries are nowadays considering the development of credit reference agencies or credit bureaus. These agencies try to collect all kinds of information of loan applicants on an aggregate level: publicly available information, previous searches, shared contributed information, aggregated information, fraud warnings, bureau-added value [235]. In most countries, credit agencies are still under development and have to obey strict legislation. However, in the U.S. and U.K. they are already well established. The most famous in the U.K. are Experian and Equifax.

Note that the conclusions found in our benchmarking study are very analogous to those found in [256] where a similar benchmarking study was conducted for expert automobile insurance fraud detection using logistic regression, C4.5, k-nearest neighbor, neural networks, LS-SVMs, naive Bayes and TANs.

3.5 Conclusions

In this chapter, we conducted a large scale benchmarking study of the performance of the classification techniques discussed in chapter 2 on 8 credit scoring data sets. The following classification techniques were considered: linear discriminant analysis, quadratic discriminant analysis, logistic regression, linear programming, (least squares) support vector machines, neural networks, naive Bayes, TAN, C4.5, C4.5rules, 10-nearest neighbor and 100-nearest neighbor. We used a training set/test set setup and the performance was quantified by the percentage correctly classified observations and the area under the receiver operating characteristic curve. In order to compute the former, we used different types of cut-off setting schemes: a cut-off of 0.5, a cut-off based on equal sample proportions and cut-offs based on marginal good-bad rates around 5:1 and 3:1. The PCCs were compared using McNemar's test and the AUCs using the test of DeLong, DeLong, and Clarke-Pearson. Average performances and rankings were also computed and compared using the Wilcoxon signed rank test.

It was concluded that many classification techniques yielded rather similar performances. The RBF LS-SVM and NN classifiers performed especially well in all cases but also simple classifiers such as LDA and LOG gave very good performances. It was also noted that in order to improve the performance of a scorecard, financial institutions should look for more powerful, discriminating inputs obtained from e.g. credit bureau agencies. An interesting topic for further research might be to investigate the additional performance benefits obtained from combining several classifiers by using voting schemes. Another avenue for future research is to repeat the same benchmarking experiment but consider a continuous dependent variable representing e.g. number of months in arrears, amount of account overdraft, etc.

In this chapter, we primarily focussed on developing scorecards that achieve a high performance in terms of PCC or AUC. However, blindly applying an optimized classification technique without thoroughly inspecting the values of its estimated parameters and reasoning is not appropriate for credit scoring. Ideally, the trained classification technique should also provide the expert with some explanation about why it classifies a particular applicant as good or bad. Capon was one of the first authors to argue that credit scoring systems should focus more on providing explanations for why customers default instead of merely trying to develop scorecards which accurately distinguish good customers from bad customers:

> *"What is needed, clearly, is a redirection of credit scoring research efforts toward development of explanatory models of credit performance and the isolation of variables bearing an explanatory relationship to credit performance" [41].*

Moreover, note that there is also often a legal obligation to give some explanation of the reason why credit is declined. Hence, for a successful adoption of the constructed scorecards into the daily credit decision environment, two properties are essential. First, the scorecards should achieve a high performance in discriminating bad customers from good customers. Furthermore, they should also be made intelligent in the sense that they provide a clear insight to the expert about how and why a certain applicant is classified as good or bad. In this chapter, we found that, amongst other, neural networks achieved a very good performance in distinguishing good customers from bad customers. However, their complex mathematical internal workings limit their comprehensibility and prohibit their practical use. In the following chapter, we will investigate how the neural network black box may be opened using rule extraction techniques that will provide the expert with a comprehensible explanation behind the classifications made by the network. In this way, we will try to build scorecards which are at the same time both powerful and understandable.

Chapter 4

Building Intelligent Systems for Credit Scoring using Neural Network Rule Extraction

In this chapter, we will investigate how neural network rule extraction techniques may be adopted to develop intelligent systems for credit scoring[1,2,3] *In chapter 3, we have seen that neural networks are amongst the best techniques for credit scoring. However, we primarily focussed there on developing networks with high predictive accuracy without trying to explain how the classifications are being made. Clearly, this plays a very pivotal role in credit-risk evaluation as the evaluator may be required to give a justification why a certain credit application is approved or rejected. Recent developments in algorithms that extract rules from trained neural networks enable us to generate explanatory classification rules that explain the decision process of the networks. The purpose of this chapter is to investigate if these neural network rule extraction techniques can generate meaningful and accurate rule sets for the credit-risk evaluation problem. Hereto, we conduct experiments on the German credit, Bene1 and Bene2 data sets. Again, our main interest lies*

[1]B. Baesens, R. Setiono, C. Mues, J. Vanthienen, Using Neural Network Rule Extraction and Decision Tables for Credit-Risk Evaluation, *Management Science*, 49(3), pp. 312-329, 2003.

[2]B. Baesens, R. Setiono, C. Mues, S. Viaene, J. Vanthienen, Building credit-risk evaluation expert systems using neural network rule extraction and decision tables, *Proceedings of the Twenty Second International Conference on Information Systems (ICIS'2001)*, New Orleans, Louisiana, USA, December, 2001.

[3]B. Baesens, C. Mues, R. Setiono, M. De Backer, J. Vanthienen, Building Intelligent Credit Scoring Systems using Decision Tables, *Proceedings of the Fifth International Conference on Enterprise Information Systems (ICEIS'2003)*, Angers, pp. 19-25, April 2003.

in finding the distinguishing characteristics between good customers and bad customers. Both the continuous and the discretized data sets will be analyzed. Techniques that will be adopted are Neurolinear [210], Neurorule [208], Trepan [53], and Nefclass [171]. The performance of these methods will be compared with the widely used C4.5 and C4.5rules algorithms [187] in terms of predictive accuracy and conciseness of the generated rule sets or decision trees. In a subsequent step of the decision support system development process, the extracted rules will be represented in an alternative way using decision tables [165, 248]. This is motivated by the fact that research in knowledge representation suggests that graphical representation formalisms can be more readily interpreted and consulted by humans than symbolic rules [199].

4.1 An Overview of Neural Network Rule Extraction

As universal approximators, neural networks can achieve significantly better predictive accuracy compared to models that are linear in the input variables. Unfortunately, an often mentioned drawback associated with using neural networks is their opacity. This refers to the fact that they do not allow formalization of the relationship between the outputs and the inputs in a user-friendly, comprehensible way. Neural networks are then commonly described as black box techniques because they generate complex mathematical models which relate the outputs to the inputs using a set of weights, biases and non-linear activation functions which are hard for humans to interpret. It is precisely this black box property that prevents them from being used as effective management science tools in real-life situations (e.g. credit-risk evaluation) where besides having accurate models, explanation of the predictions being made is essential.

In the literature, the problem of explaining the neural network predictions has been tackled by techniques that extract symbolic rules or trees from the trained networks. These neural network rule extraction techniques attempt to open up the neural network black box and generate symbolic, comprehensible descriptions with approximately the same predictive power as the neural network itself. An advantage of using neural networks as a starting point for rule extraction is that the neural network considers the contribution of the inputs towards classification as a group, while decision tree algorithms like C4.5 measure the individual contribution of the inputs one at a time as the tree is grown.

Andrews, Diederich and Tickle [5] propose a classification scheme for neural network rule extraction techniques based on the following criteria:

1. Translucency of the extraction algorithm with respect to the underlying neural network;

2. Expressive power of the extracted rules or trees;
3. Specialized training regime of the neural network;
4. Quality of the extracted rules;
5. Algorithmic complexity of the extraction algorithm.

The translucency criterion considers the technique's perception of the neural network. A decompositional approach is closely intertwined with the internal workings of the neural network. It typically starts extracting rules at the level of the individual hidden and output units by analyzing the activation values, weights and biases. On the other hand, a pedagogical algorithm considers the trained neural network as a "black box". Instead of looking at the internal structure of the network, these algorithms directly extract rules which relate the inputs and outputs of the network. These techniques typically use the trained network as an oracle to label or classify artificially generated training examples which are then used by a symbolic learning algorithm. In fact, most pedagogical algorithms lend themselves very easily to rule extraction out of other machine learning algorithms (e.g. k-nearest neighbor, support vector machines, ...).

The expressive power of the extracted rules depends on the language used to express the rules. Many types of rules have been suggested in the literature. Propositional rules are simple **If**... **Then**... expressions based on conventional propositional logic. An example of a propositional rule is:

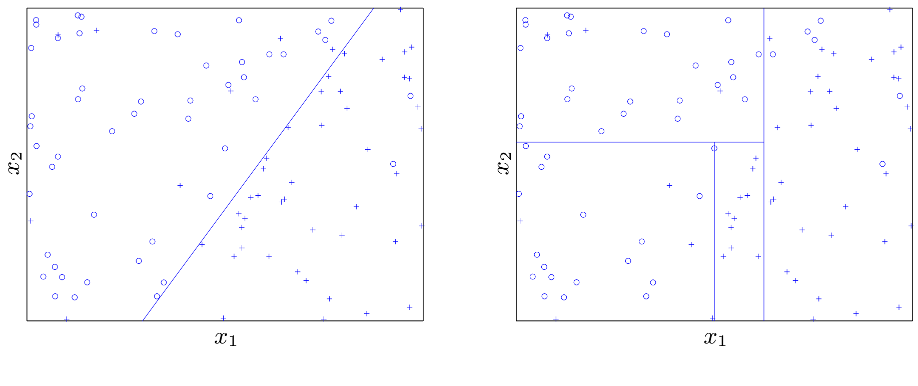
If Purpose = second hand car **And** Savings Account $\leq$ 50 Euro **Then** Applicant = bad. (4.1)

A set of propositional rules is sometimes also termed a disjunctive normal form (DNF) representation because the class concept can be expressed as a disjunction of one or more conjunctions.

An oblique rule is a propositional rule whereby each condition represents a separating hyperplane given in the form a linear inequality. An example of an oblique rule is:

If 0.84 Income + 0.32 Savings Account $\leq$ 1000 Euro **Then** Applicant = bad. (4.2)

Oblique rules allow for more powerful decision surfaces than propositional rules since the latter allow only axis-parallel decision boundaries. This is illustrated in Figure 4.1. Figure 4.1 represents a classification problem involving two classes, represented by '+' and 'o' respectively, each described by two inputs x_1 and x_2. The left hand side illustrates an oblique rule separating both classes and the right hand side a set of propositional rules inferred by e.g. C4.5 [187]. Clearly, the oblique rule provides a better separation than the set of propositional, axis-parallel, rules. Augmenting the number of training points will probably increase the number



(a) Oblique rule (b) Propositional rule

Figure 4.1: Oblique rules versus propositional rules [187].

of axis parallel decision boundaries. Hence, this example illustrates that oblique rules may provide a more powerful, concise separation than a set of propositional rules. However, this advantage has to be offset against the loss of comprehensibility since oblique rules are harder to interpret for the domain expert.

M-of-N rules are usually expressed as follows:

$$\begin{array}{l}\textbf{If}\ \{\text{at least/exactly/at most}\}\ \text{M of the N conditions}\,(\text{C1}, \text{C2}, ..., \text{CN})\\ \text{are satisfied}\ \textbf{Then}\ \text{Class} = 1.\end{array} \tag{4.3}$$

These types of rules allow one to represent complex classification concepts more succinctly than classical propositional DNF rules. Consider e.g. the Exclusive Or (XOR) problem depicted in Table 4.1. When representing the XOR problem as a

I_1	I_2	Class
0	0	0
0	1	1
1	0	1
1	1	0

Table 4.1: The XOR classification problem.

set of propositional DNF rules, one obtains:

$$\begin{array}{l}\textbf{If}\ \text{I}_1 = 0\ \textbf{And}\ \text{I}_2 = 1\ \textbf{Then}\ \text{Class} = 1,\\ \textbf{If}\ \text{I}_1 = 1\ \textbf{And}\ \text{I}_2 = 0\ \textbf{Then}\ \text{Class} = 1,\\ \text{Default Class} = 0.\end{array} \tag{4.4}$$

When using M-of-N rules, the same target concept can be represented as follows:

$$\textbf{If} \text{ exactly } 1 \text{ of } \{\mathrm{I}_1 = 1, \mathrm{I}_2 = 1\} \textbf{ Then } \text{Class} = 1, \tag{4.5}$$

which is a more concise representation than in 4.4.

All rule types considered above are crisp in the sense that their antecedent is either true or false. Fuzzy rules allow for more flexibility and are usually expressed in terms of linguistic concepts which are easier to interpret for humans. We defer the discussion on fuzzy rules to chapter 5 where we will investigate both genetic fuzzy and neurofuzzy rule extraction algorithms.

The third criterion of the schema suggested by Andrews et al. considers the portability of the rule extraction technique to various network architectures. This is closely related to the translucency criterion since pedagogical algorithms are easier to port to other networks than decompositional algorithms. Most of the latter require a specialized network architecture or training method to simplify the rule extraction process. E.g. in [208, 210] an augmented cross-entropy objective function is used to train and prune the network whereas in [6] a local cluster net is used to facilitate the rule extraction.

The quality of the extracted rule set can be expressed in various ways. Three important criteria are: accuracy, fidelity and comprehensibility. While the first and the last are obvious, the fidelity criterion measures how well the extracted rule set mimics the behavior of the neural network i.e. performs the same classifications as the neural network itself (cf. infra).

Finally, the last criterion considers the algorithmic complexity of the rule extraction algorithm. However, it has to be noted that few authors report on this issue.

We would like to add an extra criterion to the framework of Andrews et al., namely the task for which the rule extraction algorithm was developed. We hereby distinguish a classification task from a regression task. In a classification context, the target concept belongs to a specific predefined class whereas in a regression context, the output is continuous and unlimited. Recently, rule extraction algorithms have been suggested to infer rules for a regression task.

In Table 4.1, we provide an overview of some popular neural network rule extraction algorithms which have been presented in the literature. The algorithms are characterized with respect to the first two criteria of the schema of Andrews et al. and by the task for which they were originally developed.

In the following subsections, we will discuss the Neurolinear, Neurorule and Trepan extraction algorithms. The motivation for choosing these algorithms is that they have different characteristics with respect to the classification scheme suggested by Andrews et al., and that they thus tackle the extraction problem in a totally different way. To our knowledge, the performance of these algorithms has never been compared for rule or tree extraction using real-life data.

Technique	**Translucency**	**Rule expressiveness**	**Task**
Neurolinear [210]	decompositional	oblique rules	classification
Neurorule [208]	decompositional	propositional rules	classification
Trepan [53]	pedagogical	Trees with M-of-N splits	classification
Nefclass [171]	decompositional	descriptive fuzzy rules	classification
Rulex [6]	decompositional	propositional rules	classification
VIA [236]	pedagogical	propositional rules	classification
Fernn [206]	decompositional	oblique, propositional and M-of-N rules	classification
Subset [237]	decompositional	propositional rules	classification
MofN [237]	decompositional	M-of-N rules	classification
Brainne [201]	pedagogical	propositional rules	classification
Bio-Re [232]	pedagogical	propositional rules	classification
Partial-Re [232]	decompositional	propositional rules	classification
Full-Re [232]	decompositional	propositional rules	classification
Glare [102]	decompositional	propositional rules	classification
RX [203]	decompositional	propositional rules	classification
Real [52]	pedagogical	propositional and M-of-N rules	classification
Anfis [127]	decompositional	descriptive fuzzy rules	regression
Refann [207]	decompositional	propositional rules	regression
Nefprox [173]	decompositional	descriptive fuzzy rules	regression

Table 4.2: Characteristics of neural network rule extraction techniques.

4.2 Neural Network Rule Extraction using Neurolinear and Neurorule

Neurolinear and Neurorule are algorithms that extract rules from trained 3 layered feedforward neural networks. Both techniques share the following common steps [208, 210]:

1. Train a neural network to meet the prespecified accuracy requirement;
2. Remove the redundant connections in the network by pruning while maintaining its accuracy;
3. Discretize the hidden unit activation values of the pruned network by clustering;

4. Extract rules that describe the network outputs in terms of the discretized hidden unit activation values;
5. Generate rules that describe the discretized hidden unit activation values in terms of the network inputs;
6. Merge the two sets of rules generated in steps 4 and 5 to obtain a set of rules that relates the inputs and outputs of the network.

Both techniques differ in their way of preprocessing the data. Neurolinear works with continuous data which is normalized e.g. to the interval $[-1, 1]$. On the other hand, Neurorule assumes the data are discretized and represented as binary inputs using the thermometer encoding [215] for ordinal variables and dummy encoding for nominal variables.

Table 4.3 illustrates the thermometer encoding procedure for the ordinal Income variable. The continuous Income attribute is first discretized to the values

Original input	**Categorical input**	**Thermometer inputs**		
		I_1	I_2	I_3
Income $\leq$ 800 Euro	1	0	0	0
Income $>$ 800 Euro and $\leq$ 2000 Euro	2	0	0	1
Income $>$ 2000 Euro and $\leq$ 10000 Euro	3	0	1	1
Income $>$ 10000 Euro	4	1	1	1

Table 4.3: The thermometer encoding procedure for ordinal variables.

1, 2, 3 and 4. This can be done by either a discretization algorithm (e.g. the algorithm of Fayyad and Irani [83]) or according to the recommendation from the domain expert. The four values are then represented by three thermometer inputs I_1, I_2 and I_3. If I_3 is 1, this corresponds to categorical Income input ≥ 2, or original Income input $>$ 800 Euro. This encoding scheme facilitates the generation and interpretation of the propositional If-then rules as the following example illustrates.

Example 4.1
Consider the following different schemes for encoding the categorical Income attribute:

Income	Scheme 1	Scheme 2	Thermometer
	(I1,I2)	(I1)	(I1,I2,I3)
≤ 800	(0,0)	0.0	(0,0,0)
$]800, 2000]$	(0,1)	0.3	(0,0,1)
$]2000, 10000]$	(1,0)	0.6	(0,1,1)
$>$10000	(1,1)	1.0	(1,1,1)

Only the thermometer coding allows for an easy mapping of the encoded binary data back to the original attribute:

- $I3 = 0$ if and only if income is less than or equal to 800.
- $I2 = 0$ if and only if income is less than or equal to 2000.
- $I1 = 0$ if and only if income is less than or equal to 10000.

equivalently,

- $I3 = 1$ if and only if income is bigger than 800.
- $I2 = 1$ if and only if income is bigger than 2000.
- $I1 = 1$ if and only if income is bigger than 10000.

Scheme 1 does not allow such an easy interpretation of the binary encoded input. Scheme 2 cannot be used for two reasons: (a) the rule extraction algorithms work only on discrete data, (b) a sense of magnitude is implied in the encoding, for example 0.6 is 2 times 0.3, while income in $]2000, 10000]$ does not necessarily mean twice the income in $]800, 2000]$.

Neurorule assumes the nominal variables are represented by dummies. For example, when a nominal variable has 3 values, it is encoded with 2 dummy variables according to the setup shown in Table 4.4.

	I_1	I_2
Purpose=car	0	0
Purpose=real estate	0	1
Purpose=other	1	0

Table 4.4: The dummy encoding procedure for nominal variables.

Both Neurorule and Neurolinear typically start from a one-hidden layer neural network with hyperbolic tangent hidden neurons and sigmoid or linear output neurons. For a classification problem with C classes, C output neurons are used and the class is assigned to the output neuron with the highest activation value (*winner-take-all learning*). Figure 4.2 presents an example of such a network.

The network is then trained to minimize the following augmented cross-entropy error function:

$$F(\mathbf{W}, \mathbf{V}) = P(\mathbf{W}, \mathbf{V}) - \sum_{c=1}^{C} \sum_{i=1}^{N} [t_{ic} \log y_{ic} + (1 - t_{ic}) \log(1 - y_{ic})], \tag{4.6}$$

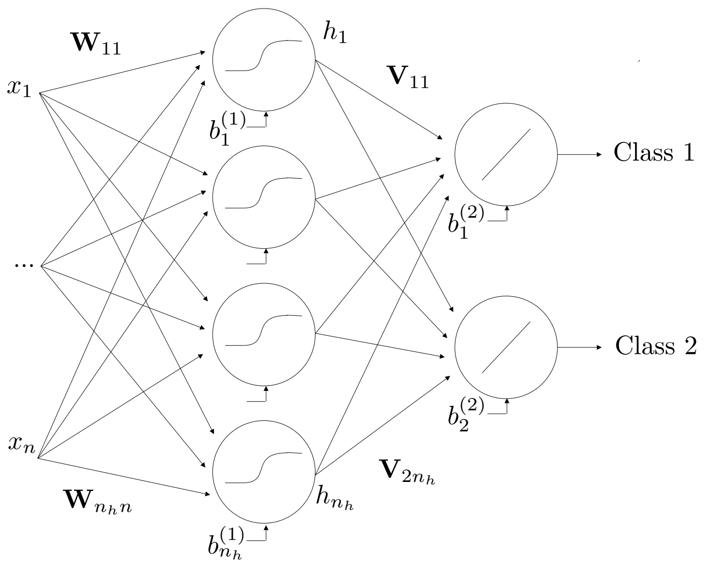


Figure 4.2: Example network used by Neurorule and Neurolinear for rule extraction.

with

$$P(\mathbf{W},\mathbf{V}) = \epsilon_1 \sum_{h=1}^{H}(\sum_{c=1}^{C}\frac{\beta\mathbf{V}_{ch}^2}{1+\beta\mathbf{V}_{ch}^2} + \sum_{j=1}^{n}\frac{\beta\mathbf{W}_{hj}^2}{1+\beta\mathbf{W}_{hj}^2}) + \epsilon_2 \sum_{h=1}^{H}(\sum_{c=1}^{C}\mathbf{V}_{ch}^2 + \sum_{j=1}^{n}\mathbf{W}_{hj}^2), \tag{4.7}$$

where C is the number of classes, N the number of data points, n the number of inputs, H the number of hidden nodes, t_{ic} is 1 if observation i belongs to class c and 0 otherwise, y_{ic} is the neural network output for the c^{th} class of the i^{th} observation, $\mathbf{W}_{hj}$ is the weight connecting input node j with hidden node h, $\mathbf{V}_{ch}$ is the weight connecting hidden node h with the output class node c, and ϵ_1, ϵ_2 and β are positive parameters.

The rationale behind the cross-entropy error function has been discussed in subsection 2.1.1 of chapter 2. The penalty function $P(\mathbf{W},\mathbf{V})$ is added to the cost function $F(\mathbf{W},\mathbf{V})$ of the network to encourage weight decay. Since $f(w) = \frac{w^2}{1+w^2}$ approaches 0 when w is small and 1 when w is large, the first term of $P(\mathbf{W},\mathbf{V})$ approximates the number of relevant, non-zero weights in the network. The β parameter is then added to control how fast the irrelevant weights converge to zero. The larger the β parameter, the faster the irrelevant weights will converge to zero. The second part of $P(\mathbf{W},\mathbf{V})$ additionally prevents these weights from taking on excessive values. The parameters ϵ_1 and ϵ_2 then reflect the relative importance of the accuracy of the neural network versus its complexity. Typical values for these parameters are: β=10, ϵ_1=10^{-1} and ϵ_2=10^{-5} [204, 208]. The cost function $F(\mathbf{W}, \mathbf{V})$ is minimized using the BFGS method which is a modified Quasi-Newton algorithm [23, 60]. This algorithm converges much faster than the standard backpropagation algorithm and the total error decreases after each iteration step which is not necessarily the case in the backpropagation algorithm [202].

Determining the optimal number of hidden neurons is not a trivial task. In the literature, two approaches have been suggested to tackle this problem. A growing strategy starts from an empty network and gradually adds hidden neurons to improve the classification accuracy. On the other hand, a pruning strategy starts from an oversized network and removes the irrelevant connections. When all connections to a hidden neuron have been removed, it can be pruned. The latter strategy is followed by Neurolinear and Neurorule. The inclusion of the term $P(\mathbf{W}, \mathbf{V})$ into the objective function $F(\mathbf{W}, \mathbf{V})$ of the network allows it to efficiently remove connections based upon the magnitude of the weights. In [204], two criteria were proposed:

- If $\omega_{ij} = \max_{k=1,2} |\mathbf{W}_{ji} \times \mathbf{V}_{kj}| < \eta$, then the connection from input unit i to hidden unit j is removed from the network.
- If $|\mathbf{V}_{kj}| < \eta$, then the connection from hidden unit j to output unit k is removed.

The prescribed value of η is given in [204]. Using this value, it is guaranteed that removal of a connection that satisfies one of the two criteria will not affect the accuracy of the network on the training data set. Note that this pruning step plays an important role in both rule extraction algorithms since it will facilitate the extraction of a compact, parsimonious rule set. After having removed one or more connections, the network is retrained and inspected for further pruning.

Once a trained and pruned network has been obtained, the activation values of all hidden neurons are clustered. Clustering makes it easier to extract rules because:

- The number of discrete representations of the data is reduced significantly. Consider a data set with thousands of samples such as the Bene1 data set and suppose the clustering process groups the activation values into two subintervals: those in $[-1, \alpha)$ and those in $[\alpha, 1]$. This makes the rule extraction process trivial: those samples with activation values in the first subinterval, predict Class 1. Samples in the second subinterval, predict Class 2.
- In general, from a network with H hidden units with I_h subintervals for hidden units $h = 1, 2, \ldots H$, there will be at most $S = I_1 \times I_2 ... \times I_H$ unique samples, and S is usually a very small fraction of the original number of samples.

In the case of hyperbolic tangent hidden neurons, the activation values lie in the interval $[-1, 1]$. A simple greedy clustering algorithm then starts by sorting all these hidden activation values in increasing order [211]. Adjacent values are then merged into a unique discretized value as long as the class labels of the corresponding observations do not conflict. The merging process hereby first considers the pair of hidden activation values with the shortest distance in between. Another discretization algorithm is the Chi2 algorithm which is an improved and automated version of the ChiMerge algorithm [137] and makes use of the χ^2 test statistic to merge the hidden activation values [152].

In step 4 of Neurolinear and Neurorule, a new data set is composed consisting of the discretized hidden unit activation values and the class labels of the corresponding observations. Duplicate observations are removed and rules are inferred relating the class labels to the clustered hidden unit activation values. This can be done using an automated rule induction algorithm such as X2R [153] or manually when the pruned network has only a few hidden neurons and inputs. Note that steps 3 and 4 can be done simultaneously by C4.5(rules) since the latter can work with both discretized and continuous data [187].

In the last two steps of both rule extraction algorithms, the rules of step 4 are translated in terms of the original inputs. First, the rules are generated describing the discretized hidden unit activation values in terms of the original inputs. This rule set is then merged with that of step 4 by replacing the conditions of the latter with those of the former. For Neurolinear, this process is fairly straightforward as the following example illustrates. Suppose we have a hidden unit h with two incoming weights w_1 and w_2 corresponding to the inputs I_1 and I_2. Furthermore, for the sake of simplicity, suppose the activation values of h have been discretized into the intervals $[-1, a[$ and $[a, 1]$ and the following rule has been inferred:

$$\textbf{If} \text{ discretized activation } h = 1 \textbf{ Then } \text{Class} = 1. \tag{4.8}$$

Translating the rule in terms of the original inputs, we have:

$$\textbf{If } tanh(w_1 I_1 + w_2 I_2) < a \textbf{ Then } \text{Class} = 1, \tag{4.9}$$

or

$$\textbf{If } (w_1 I_1 + w_2 I_2) < tanh^{-1}(a) \textbf{ Then } \text{Class} = 1, \tag{4.10}$$

with

$$tanh^{-1}(x) = \frac{ln(\frac{1+x}{1-x})}{2}. \tag{4.11}$$

The above rule then represents an oblique classification rule. In the case of Neurorule, one might again use an automated rule induction algorithm (e.g. X2R, C4.5) to relate the discretized hidden unit activation values to the inputs.

In [114], both Neurolinear and Neurorule were applied to the diagnosis of hepatobiliary disorders. It was concluded that the rules generated by Neurolinear were slightly more accurate and concise than the rules generated by Neurorule. In [205], Neurorule was applied for the diagnosis of breast cancer and in [211] to detect the characteristics of organizations adopting information technology.

4.3 Neural Network Tree Extraction using Trepan

Trepan was first introduced in [50, 53]. It is a pedagogical tree extraction algorithm extracting decision trees from trained neural networks with arbitrary architecture. Like in most decision tree algorithms [33, 187], Trepan grows a tree by recursive partitioning. However, it hereby uses a best-first expansion instead of a depth-first strategy. At each step, a queue of leaves is further expanded into sub-trees until a stopping criterion is met. Another crucial difference with existing decision tree induction algorithms is that the latter have only a limited set of training observations available. Hence, these algorithms typically suffer from having fewer and fewer training observations available for deciding upon the splits or leaf node class labels at lower levels of the tree. On the other hand, the primary goal of neural network rule extraction is to mimic the behavior of the trained neural network. Hence, instead of using the original training observations, Trepan first relabels them according to the classifications made by the network. The relabelled training data set is then used to initiate the tree growing process. Furthermore, Trepan can also enrich the training data with additional training instances which are then also labelled (classified) by the neural network itself. The network is thus used as an oracle to answer class membership queries about artificially generated data points. This way, it can be assured that each node split or leave node class decision is based upon at least S_{min} data points where S_{min} is a user defined parameter. In other words, if a node has only m training data points available and $m < S_{min}$, then $S_{min} - m$ data points are additionally generated and labelled by the network. This process is often referred to as *active learning*.

Generating these additional data points is by no means a trivial task. First of all, care should be taken that the generated data instances satisfy all constraints (conditions) that lie from the root of the tree to the node under consideration. Given these constraints one approach might be to sample the data instances uniformly. However, a better alternative would be to take into account the distribution of the data. This is the approach followed by Trepan. More specifically, at each node of the tree, Trepan estimates the marginal distribution of each input. For a discrete valued input, Trepan simply uses the empirical frequencies of the various values whereas for a continuous input x, a kernel density estimation method is used to model the probability distribution $f(x)$ as follows [214]:

$$f(x) = \frac{1}{m}\sum_{j}^{m}[\frac{1}{\sqrt{2\pi}}\exp^{-(\frac{x-\mu_j}{2\sigma})^2}], \tag{4.12}$$

whereby m is the number of training examples used in the estimate, μ_j is the value of the input for the j^{th} example, and σ is the width of the Gaussian kernel. Trepan sets σ to $\frac{1}{\sqrt{m}}$. One important shortcoming of this procedure is that one estimates marginal distributions instead of a joint distribution and thus the dependencies between the inputs are not properly taken into account. Trepan partially overcomes this by estimating separate distributions for each node of the tree hereby using only the training instances that reach the given node. This allows it to capture some of the conditional dependencies since the estimated distributions are conditionally dependent upon the outcome of the tests between the root node and the given node. However, since this density estimation procedure is solely based on original training data, it follows that the estimates become less reliable for lower levels in the tree since they are based on less training data. To counter this, Trepan uses a statistical test to see if the local distributions at a given node are significantly different from the distributions at the ancestor node. If not, the distributions of the ancestor node are used.

Trepan allows splits with *at least M-of-N* type of tests. Note that the test *at least 2 of {C1,C2,C3}* is logically equivalent to *(C1 And C2) Or (C1 And C3) Or (C2 and C3).* These M-of-N splits are constructed by using a heuristic search procedure. First, the best binary split is selected according to the information gain criterion of Equation 2.27. For discrete inputs this is based upon their distinct values and for continuous inputs, the values are first sorted and the midpoints between adjacent values of different classes are considered as potential thresholds. The best binary test then serves as a seed for the M-of-N search process which uses the following operators [167]:

- M-of-N+1: Add a new condition to the set.
  E.g. 2 of {C1,C2} becomes 2 of {C1,C2,C3}.
- M+1-of-N+1: Add a new condition to the set and augment the threshold.
  E.g. 2 of {C1,C2,C3} becomes 3 of {C1,C2,C3,C4}.

The heuristic search uses a beam-search method with a beam width of two meaning that at each point the best two splits are retained for further examination. Again, the information gain criterion is used to evaluate the splits. Finally, once an M-of-N test has been constructed, Trepan tries to simplify it and investigates if conditions can be dropped and/or M can be reduced without significantly degrading the information gain.

Trepan uses one local and two global criteria to decide when to stop growing the tree. For the local stopping criterion, Trepan constructs a confidence interval around p_c which is the proportion of instances belonging to the most common class at the node under consideration. The node becomes a leaf when $prob(p_c < 1-\epsilon) < \alpha$ whereby α is the significance level and ϵ specifies how tight the confidence interval around p_c must be. Both values are set to 0.01 by default.

The first global criterion specifies a maximum on the number of internal nodes of the tree and can be specified in advance by the user. Trees with a small number of internal nodes are more comprehensible than large trees. The second global criterion involves the use of a validation set, together with the size limit, to decide upon the tree to return. This validation set is used to monitor the fidelity of each of the trees in the construction sequence and the tree with the highest fidelity is returned.

In [51], Trepan was applied to extract trees for exchange rate prediction. It was shown that Trepan is able to extract compact and powerful decision trees when compared to the C4.5 and the enhanced ID2-of-3 algorithm.

4.4 Neural Network Rule Extraction for Credit Scoring

4.4.1 Experimental Setup

In this section, the neural network rule extraction techniques described in the previous sections are applied to the German credit, Bene1 and Bene2 credit scoring data sets discussed in chapter 3 [10, 11]. Again, each data set is randomly split into two-thirds training set and one-third test set. The neural networks are trained and rules are extracted using the training set. The test set is then used to assess the predictive power of the trained networks and the extracted rule sets or trees. Both the continuous and the discretized data sets will be investigated. The continuous data sets will be analyzed using Neurolinear, Trepan and Nefclass. For the discretized data sets we will use Neurorule, Trepan and Nefclass. Nefclass is a neurofuzzy classifier generating fuzzy if-then rules. We defer the discussion on the functioning of Nefclass to chapter 5 where also more experimental evidence will be reported on fuzzy classification. We will also include C4.5 and C4.5rules as a benchmark to compare the results of the rule extraction algorithms. As in chapter 3, we set the confidence level for the pruning strategy to 25% which is the value that is commonly used in the literature.

All algorithms will be evaluated by their classification accuracy as measured by the percentage correctly classified (PCC) observations and their complexity. Since our main purpose is to develop intelligent credit scoring systems that are both comprehensible and user-friendly, it is obvious that simple, concise rule sets and trees are to be preferred. Hence, we will also take into account the complexity of the generated rules or trees as a performance measure. The complexity will be quantified by looking at the number of generated rules or the number of leave nodes and total number of nodes for the Trepan trees. Note that the total number of nodes of a tree is the sum of the number of internal nodes and the number of leave nodes.

Since the primary goal of neural network rule extraction is to mimic the decision process of the trained neural network, we will also measure how well the extracted rule set or tree models the behavior of the network. For this purpose, we will also measure the fidelity of the extraction techniques which is defined as the percentage of observations that the extraction algorithm classifies in the same way as the neural network. More specifically, if we represent the classifications of the neural network and the extraction algorithm as the confusion matrix of Table 4.5, then

		NN classification	
		good	bad
Rule extraction	good	a	b
classification	bad	c	d

Table 4.5: The fidelity measure.

the fidelity is defined as follows:

$$\text{fidelity} = \frac{a+d}{a+b+c+d}. \tag{4.13}$$

For the Neurolinear and Neurorule analyses, we use two output units with linear or logistic activation functions and the class is assigned to the output neuron with the highest activation value (winner-takes-all). A hyperbolic tangent activation function is used in the hidden layer.

Following Craven and Shavlik [53], we set the S_{min} parameter for the Trepan analyses to 1000 meaning that at least 1000 observations are considered before deciding upon each split or leave node class label. The maximum tree size is set to 15 which is the size of a complete binary tree of depth four.

Since Trepan is a pedagogical tree extraction algorithm, we can apply it to any trained neural network with arbitrary architecture. Hence, we will apply Trepan to the same networks that were trained and pruned by Neurolinear and Neurorule. This will allow us to make a fair comparison between a pedagogical and a decompositional neural network rule extraction method.

For Nefclass, we will experiment with triangular, trapezoidal and bell-shaped membership functions and use 2, 4 or 6 fuzzy sets per variable. We will also use both *Best rule learning* and *Best per Class rule learning* with a maximum of 100 fuzzy rules.

4.4.2 Results for the Continuous Data Sets

Table 4.6 presents the results of applying the rule extraction methods to the continuous data sets. Before the neural networks are trained for rule extraction using

Neurolinear, all inputs x_i, $i = 1, ..., n$ are scaled to the interval $[-1, 1]$ in the following way:

$$x_i^{new} = 2[\frac{x_i^{old} - \min(x_i)}{\max(x_i) - \min(x_i)}] - 1. \tag{4.14}$$

As explained in section 4.2, Neurolinear typically starts from a large, oversized network and then prunes the irrelevant connections. These pruned neural networks which are used by both Neurolinear and Trepan have 1 hidden unit for the German credit and Bene2 data set and 2 hidden units for the Bene1 data set. This again clearly illustrates that credit scoring is a problem which might be best approached using simple classification models such as a neural network with one (or two) hidden unit(s) and two output units. The pruning procedure indicates that there is obviously no need to model more complex non-linearities by using more hidden neurons. The pruned networks had 16 inputs for the German credit data set, 17 inputs for the Bene1 data set and 23 inputs for the Bene2 data set.

Neurolinear obtained 100% test set fidelity for the German credit and Bene2 data set and 99.9% test set fidelity for the Bene1 data set. The test set fidelity of Trepan with respect to the neural networks which were also used by Neurolinear is 91.31%, 87.60% and 85.31% for the German credit, Bene1 and Bene2 data set, respectively. This clearly indicates that Neurolinear was able to extract rule sets which better reflect the decision process of the trained neural networks than the trees inferred by Trepan.

It can be observed from Table 4.6 that the rules and trees extracted by Neurolinear and Trepan are both powerful and very concise when compared to the rules and trees inferred by C4.5rules and C4.5. Neurolinear yields the best absolute test set performance for all three data sets with a maximum of three oblique rules for the Bene1 data set. For the German credit data set, Neurolinear performed significantly better than C4.5rules according to McNemar's test at the 1% level. For the Bene1 and Bene2 data sets, the performance of Neurolinear was not significantly different from C4.5rules at the 5% level. Neurolinear obtained a significantly better performance than Nefclass on all three data sets. Nefclass was never able to extract compact and powerful fuzzy rule sets for any of the data sets. We found that its performance in terms of classification accuracy was very dependent upon the specific parameter setting. Note also that for all three data sets, Trepan obtained a better classification accuracy than C4.5 with much fewer leaves and nodes.

Figure 4.3 and 4.4 depict the oblique rules that were extracted by Neurolinear for the German credit and Bene1 data sets. Obviously, although the rules perfectly mimic the decision process of the corresponding neural networks, their comprehensive value is rather limited. They are basically mathematical expressions which represent piece-wise linear discriminant functions. Hence, their usefulness for building intelligent, user-friendly and comprehensible credit scoring systems can be questioned.

Data set	**Method**	$\mathbf{PCC_{train}}$	$\mathbf{PCC_{test}}$	**Complexity**
German credit	C4.5	82.58	70.96	37 leaves, 59 nodes
	C4.5rules	81.53	70.66	13 propositional rules
	Pruned NN	80.78	77.25	16 inputs
	Neurolinear	80.93	77.25	2 oblique rules
	Trepan	75.97	73.35	6 leaves, 11 nodes
	Nefclass	71.20	70.36	16 fuzzy rules
Bene1	C4.5	89.91	68.68	168 leaves, 335 nodes
	C4.5rules	78.63	70.80	21 propositional rules
	Pruned NN	77.33	72.62	17 inputs
	Neurolinear	77.43	72.72	3 oblique rules
	Trepan	73.29	70.60	12 leaves, 23 nodes
	Nefclass	67.53	66.19	8 fuzzy rules
Bene2	C4.5	90.24	70.09	849 leaves, 1161 nodes
	C4.5rules	77.61	73.00	30 propositional rules
	Pruned NN	76.05	73.51	23 inputs
	Neurolinear	76.05	73.51	2 oblique rules
	Trepan	73.36	71.84	4 leaves, 7 nodes
	Nefclass	69.43	69.25	2 fuzzy rules

Table 4.6: Neural network rule extraction results for the continuous data sets.

Data set	**Method**	$\mathbf{Fid_{train}}$	$\mathbf{Fid_{test}}$
German credit	Neurolinear	100	100
	Trepan	89.78	91.31
Bene1	Neurolinear	99.81	99.90
	Trepan	87.51	87.60
Bene2	Neurolinear	100	100
	Trepan	82.94	85.31

Table 4.7: Fidelity rates of extraction techniques.

If [-24.59(Checking account)+ 29.66(Term)-16.45(Credit history)
-3.66(Purpose)-18.69(Savings account)+9.29(Installment rate)
-18.74(Personal status)+6.19(Property) -10.03(Age)
-9.36 (Other installment plans)-11.51(Housing)+7.15(Existing credits)
+16.68(Job)+2.046(Number of dependents)-4.54(Telephone)
-8.29(Foreign worker)] $\leq$ 0.15
Then Applicant=good
Else Applicant=bad

Figure 4.3: Oblique rules extracted by Neurolinear for German credit.

If [-12.83(Amount on purchase invoice)+13.36(Percentage of financial burden) +31.33(Term)-0.93(Private or professional loan))-35.40(Savings account) -5.86(Other loan expenses)+10.69(Profession)+10.84(Number of years since last house move)+3.03(Code of regular saver)+6.68(Property)-6.02(Existing credit info)-13.78(Number of years client)-2.12(Number of years since last loan) -10.38(Number of mortgages)+68.45(Pawn)-5.23(Employment status) -5.50(Title/salutation)] $\leq$ 0.31
Then Applicant=good

If [19.39(Amount on purchase invoice)+32.57(Percentage of financial burden) -5.19(Term)-16.75(Private or professional loan)-27.96(Savings account) +7.58(Other loan expenses)-13.98(Profession)-8.57(Number of years since last house move)+6.30(Code of regular saver)+3.96(Property) -9.07(Existing credit info)-0.51(Number of years client) -5.76(Number of years since last loan) +0.14(Number of mortgages)+0.15(Pawn)+1.14(Employment status) +15.03(Title/salutation)]$\leq$ -0.25
Then Applicant=good

Default Class: Applicant=bad

Figure 4.4: Oblique rules extracted by Neurolinear for Bene1.

4.4.3 Results for the Discretized Data Sets

After discretization using the method of Fayyad and Irani, 15 inputs remained for the German credit data set, 21 inputs for the Bene1 data set and 29 inputs for the Bene2 data set. When representing these inputs using the thermometer and dummy encoding, we ended up with 45 binary inputs for the German credit data set, 45 binary inputs for the Bene1 data set and 105 inputs for the Bene2 data set. We then trained and pruned the neural networks for rule extraction using Neurorule and tree extraction using Trepan. All these neural networks have hyperbolic tangent activation functions in the hidden layer and linear output activation functions.

Figure 4.5 depicts the neural network that was trained and pruned for the Bene1 data set. Only 1 hidden unit was needed with a hyperbolic tangent transfer function. All inputs are binary, e.g. the first input is 1 if Term > 12 Months and 0 otherwise. Note that according to the pruning algorithm, no bias was needed to the hidden neuron for the Bene1 data set. Of the 45 binary inputs, 37 were pruned leaving only 8 binary inputs in the neural network. This corresponds to 7 of the original inputs depicted in Table D.1 of the Appendix because the nominal Purpose input has two corresponding binary inputs in the pruned network (Purpose= cash provisioning and Purpose=second hand car).

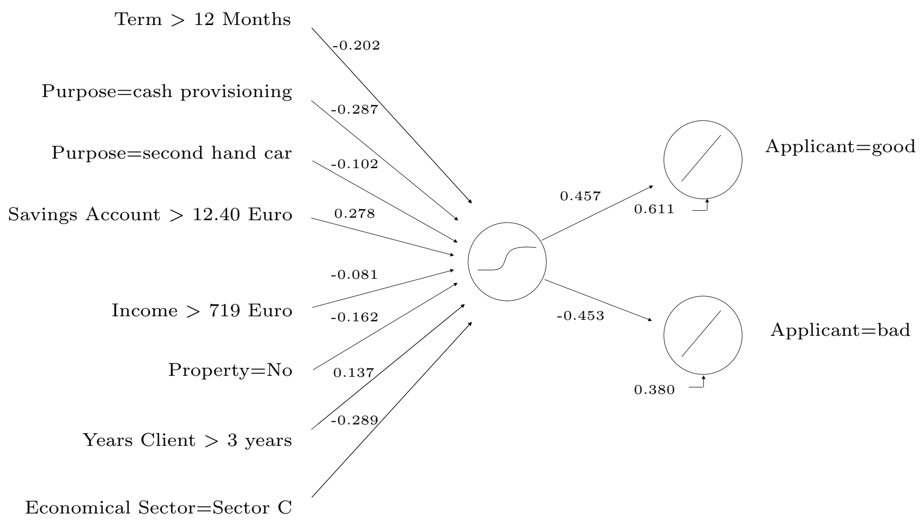


Figure 4.5: Neural network trained and pruned for Bene1.

The network trained and pruned for Bene2 had 1 hidden neuron with again no bias input. Starting from 105 binary inputs, the pruning procedure removed 97 of them and the remaining 8 corresponded to 7 of the original inputs. The network for German credit had also only 1 hidden neuron but with a bias input. The binarized German credit data set consists of 45 inputs of which 13 are retained, corresponding to 6 of the original inputs of Table C in the Appendix.

Three things are worth mentioning here. First of all, observe how the pruned networks for all three data sets have only 1 hidden neuron. These networks are thus only marginally different from an ordinary logistic regression model. This clearly confirms our earlier finding that, also for the discretized data sets, simple classification models yield good performance for credit scoring. Furthermore, since all networks have only 1 hidden neuron, the rule extraction process by Neurorule can also be simplified. If we would cluster the hidden unit activation values by sorting them, we would find two clusters corresponding to the two output classes. Hence, instead of generating the rules relating the outputs to the clustered hidden unit activation values and merge them with the rules expressing the clustered hidden unit activation values in terms of the inputs, we can generate the rules relating the outputs to the inputs directly by using C4.5rules. Finally, also notice how the binary representation allows to prune more inputs than with the continuous data sets. This will off course facilitate the generation of a compact set of rules or tree.

Table 4.8 presents the performance and complexity of C4.5, C4.5rules, the pruned NN, Neurorule, Trepan and Nefclass on the discretized credit scoring data sets. It is important to remark here that the discretization process introduces non-linear effects.

Data set	**Method**	$\mathbf{PCC_{train}}$	$\mathbf{PCC_{test}}$	**Complexity**
German credit	C4.5	80.63	71.56	38 leaves, 54 nodes
	C4.5rules	81.38	74.25	17 propositional rules
	Pruned NN	75.53	77.84	6 inputs
	Neurorule	75.83	77.25	4 propositional rules
	Trepan	75.37	73.95	11 leaves, 21 nodes
	Nefclass	73.57	73.65	14 fuzzy rules
Bene1	C4.5	77.76	70.03	77 leaves, 114 nodes
	C4.5rules	76.70	70.12	17 propositional rules
	Pruned NN	73.05	71.85	7 inputs
	Neurorule	73.05	71.85	6 propositional rules
	Trepan	73.05	71.85	11 leaves, 21 nodes
	Nefclass	68.97	67.24	8 fuzzy rules
Bene2	C4.5	82.80	73.09	438 leaves, 578 nodes
	C4.5rules	77.76	73.51	27 propositional rules
	Pruned NN	74.15	74.09	7 inputs
	Neurorule	74.27	74.13	7 propositional rules
	Trepan	74.15	74.01	9 leaves, 17 nodes
	Nefclass	70.06	69.80	4 fuzzy rules

Table 4.8: Neural network rule extraction results for the discretized data sets.

Data set	**Method**	$\mathbf{Fid_{train}}$	$\mathbf{Fid_{test}}$
German credit	Neurorule	99.70	98.80
	Trepan	94.07	93.11
Bene1	Neurorule	100	100
	Trepan	100	100
Bene2	Neurorule	99.71	99.79
	Trepan	99.91	99.83

Table 4.9: Fidelity rates of extraction techniques.

When comparing Table 4.8 with Table 4.6 it can be seen that the test set performance in most cases augments and that the discretization process did not cause any loss of predictive power of the inputs.

For the German credit data set, Neurorule did not perform significantly better than C4.5rules at the 5% level according to McNemar's test. However, Neurorule extracted only 4 propositional rules which is very compact when compared to the 17 propositional rules inferred by C4.5rules. The Trepan tree obtained a better classification accuracy than C4.5 with fewer leaves and nodes. Also Nefclass obtained a good classification accuracy but it needed 14 fuzzy rules. The test set fidelity of Neurorule is 98.80% whereas Trepan obtained 93.11% test set fidelity which indicates that Neurorule mimics the decision process of the network better than Trepan.

For the Bene1 data set, Neurorule performed significantly better than C4.5rules at the 5% level. Besides the gain in performance, Neurorule also uses only 6 propositional rules whereas C4.5rules uses 17 propositional rules. The rule set inferred by Neurorule obtained 100% test set fidelity with respect to the pruned neural network from which it was derived. Trepan gave better performance than C4.5. Again, the tree was a lot more compact consisting of only 11 leaves and 21 nodes. The Trepan tree also achieved 100% test set fidelity with respect to the pruned neural network. The high fidelity rates of Neurorule and Trepan indicate that they were able to accurately approximate the decision process of the trained and pruned neural network. Nefclass yielded a maximum test set accuracy of 67.24% with 8 fuzzy rules which is rather bad compared to the other extraction algorithms.

For the Bene2 data set, the performance difference between Neurorule and C4.5rules is not statistically significant at the 5% level using McNemar's test. However, the rule set extracted by Neurorule consists of only 7 propositional rules which is a lot more compact than the 27 propositional rules induced by C4.5rules. Note that the rules extracted by Neurorule yielded a better classification accuracy than the network from which they were derived resulting in a test set fidelity of 99.79%. The tree inferred by Trepan has a very good performance and was again compact when compared to the C4.5 tree. Trepan achieved 99.83% test set fidelity. Again, Nefclass was not able to infer a compact and powerful fuzzy rule set.

Figure 4.6 and Figure 4.7 represent the rules extracted by Neurorule for the German credit and Bene1 data sets whereas Figure 4.8 and Figure 4.9 represent the extracted Trepan trees for both data sets. Notice that both Trepan trees make extensively use of the M-of-N type of splits. Although these splits allow to make powerful split decisions, their comprehensive value is rather limited. It is very difficult to comprehend a Trepan tree and get a thorough insight into how the inputs affect the classification decision when there are M-of-N type of splits present. On the other hand, when looking at the rules extracted by Neurorule, it becomes clear that these propositional rules are easy to interpret and understand. These propositional rules are more comprehensible than the oblique rules extracted by Neurolinear or the trees with M-of-N splits extracted by Trepan. However, while propositional rules are an intuitive and well-known formalism to represent knowledge, they are not necessarily the most suitable representation in terms of structure and efficiency of use in every day business practice and decision-making. Recent research in knowledge representation suggests that graphical representation formalisms can be more readily interpreted and consulted by humans than a set of symbolic propositional if-then rules [199]. In the following section, we will discuss how the extracted sets of rules may be transformed into decision tables which facilitate the efficient classification of applicants by the credit-risk manager.

If (Checking account $\neq$ 4) **And** (Checking account $\neq$ 3) **And** (Term = 1)
And (Credit history $\neq$ 4) **And** (Credit history $\neq$ 3)
And (Credit history $\neq$ 2) **And** (Purpose $\neq$ 8)
Then Applicant = bad

If (Checking account $\neq$ 4) **And** (Checking account $\neq$ 3)
And (Credit history $\neq$ 4) **And** (Credit history $\neq$ 3)
And (Credit history $\neq$ 2) **And** (Term = 2)
Then Applicant = bad

If (Checking account $\neq$ 4) **And** (Checking account $\neq$ 3)
And (Credit history $\neq$ 4) **And** (Purpose $\neq$ 5) **And** (Purpose $\neq$ 1)
And (Savings account $\neq$ 5) **And** (Savings account $\neq$ 4)
And (Other parties $\neq$ 3) **And** (Term = 2)
Then Applicant = bad

Default class: Applicant = good

Figure 4.6: Rules Extracted by Neurorule for German credit.

If Term > 12 months **And** Purpose = cash provisioning **And** Savings account $\leq$ 12.40 Euro **And** Years client $\leq$ 3 **Then** Applicant = bad

If Term > 12 months **And** Purpose = cash provisioning **And** Owns property = No **And** Savings account $\leq$ 12.40 Euro **Then** Applicant = bad

If Purpose = cash provisioning **And** Income > 719 Euro **And** Owns property = No **And** Savings account $\leq$ 12.40 Euro **And** Years client $\leq$ 3 **Then** Applicant = bad

If Purpose = second hand car **And** Income > 719 Euro **And** Owns property = No **And** Savings account $\leq$ 12.40 Euro **And** Years client $\leq$ 3 **Then** Applicant = bad

If Savings account $\leq$ 12.40 Euro **And** Economical sector = Sector C **Then** Applicant = bad

Default class: Applicant = good

Figure 4.7: Rules Extracted by Neurorule for Bene1.

3 of {Credit history $\neq$ 4, Term = 2, Checking account $\neq$ 4}:
| **2 of** {Credit history = 2, Savings account = 5, Purpose = 1}: Applicant = good
| **Not 2 of** {Credit history = 2, Savings account = 5, Purpose = 1}:
| | Checking account $\neq$ 3:
| | | Other parties $\neq$ 3:
| | | | **1 of** {Credit history = 3, Savings account = 4}: Applicant = good
| | | | **Not 1 of** {Credit history = 3, Savings account = 4}:
| | | | | Purpose $\neq$ 5:
| | | | | | Credit history $\neq$ 2:
| | | | | | | Savings account $\neq$ 3:
| | | | | | | | Savings account $\neq$ 5:
| | | | | | | | | Purpose $\neq$ 1: Applicant = bad
| | | | | | | | | Purpose = 1: Applicant = good
| | | | | | | | Savings account = 5: Applicant = good
| | | | | | | Savings account = 3: Applicant = good
| | | | | | Credit history = 2: Applicant = bad
| | | | | Purpose = 5: Applicant = good
| | | Other parties = 3: Applicant = good
| | Checking account = 3: Applicant = good
Not 3 of {Credit history $\neq$ 4, Term = 2, Checking account $\neq$ 4}: Applicant = good

Figure 4.8: Tree extracted by Trepan for German credit.

2 of{purpose $\neq$ car, Savings account > 12.40 Euro, purpose $\neq$ cash}:
| Economical sector $\neq$ C: Applicant = good
| Economical sector = C:
| | Savings account $\leq$ 12.40Euro: Applicant = bad
| | Savings account > 12.40 Euro: Applicant = good
Not 2 of {purpose $\neq$ car, Savings account > 12.40 Euro, purpose $\neq$ cash}:
| **3 of** {Economical sector $\neq$ C, Term $\leq$ 12, Property = Yes, Years client > 3}:
| | Applicant = good
| **Not 3 of** {Economical sector $\neq$ C, Term $\leq$ 12, Property = Yes, Years client > 3}:
| | purpose $\neq$ cash:
| | | Income $\leq$ 719 Euro: Applicant = good
| | | Income > 719 Euro:
| | | | Property = Yes: Applicant = good
| | | | Property = No:
| | | | | Years client $\leq$ 3: Applicant = bad
| | | | | Years client > 3: Applicant = good
| | purpose = cash:
| | | Income $\leq$ 719 Euro:
| | | | Term $\leq$ 12 Months: Applicant = good
| | | | Term > 12 Months: Applicant = bad
| | | Income > 719 Euro: Applicant = bad

Figure 4.9: Tree extracted by Trepan for Bene1.

4.5 Visualizing the Extracted Rule Sets using Decision Tables

Decision tables provide an alternative way of representing data mining knowledge extracted by e.g. neural network rule extraction in a user-friendly way [260]. Decision tables (DTs) are a tabular representation used to describe and analyze decision situations (e.g. credit-risk evaluation), where the state of a number of conditions jointly determines the execution of a set of actions. In our neural network rule extraction context, the conditions correspond to the antecedents of the rules whereas the actions correspond to the outcome classes (Applicant = good or bad). A DT consists of four quadrants, separated by double-lines, both horizontally and vertically (cf. Figure 4.10). The horizontal line divides the table into a condition part (above) and an action part (below). The vertical line separates subjects (left) from entries (right). The condition subjects are the criteria that

condition subjects	condition entries
action subjects	action entries

Figure 4.10: DT quadrants.

are relevant to the decision making process. They represent the attributes of the rule antecedents about which information is needed to classify a given applicant as good or bad. The action subjects describe the possible outcomes of the decision making process (i.e., the classes of the classification problem). Each condition entry describes a relevant subset of values (called a state) for a given condition subject (attribute), or contains a dash symbol ('-') if its value is irrelevant within the context of that column. Subsequently, every action entry holds a value assigned to the corresponding action subject (class). True, false and unknown action values are typically abbreviated by '$\times$', '-', and '.', respectively. Every column in the entry part of the DT thus comprises a classification rule, indicating what action(s) apply to a certain combination of condition states. If each column only contains simple states (no contracted or irrelevant entries), the table is called an expanded DT, whereas otherwise the table is called a contracted DT.

Table contraction can be achieved by combining logically adjacent (groups of) columns that lead to the same action configuration. For ease of legibility, only contractions are allowed that maintain a lexicographical column ordering, i.e., in which the entries at lower rows alternate before the entries above them; see Figure 4.11 (Figure 4.12) for an example of an (un)ordered DT, respectively. As a result of this ordering restriction, a tree structure emerges in the condition entry part of the DT, which lends itself very well to a top-down evaluation procedure: starting at the first row, and then working one's way down the table by choosing from the relevant condition states, one safely arrives at the prescribed action (class) for a given case. The number of columns in the contracted table can be further minimized by changing the order of the condition rows. It is obvious that a DT with a minimal number of columns is to be preferred since it provides a more parsimonious and comprehensible representation of the extracted knowledge than an expanded DT. This is illustrated in Figure 4.11.

1. Owns property?	yes				no			
2. Years client	≤ 3		>3		≤ 3		>3	
3. Savings amount	low	high	low	high	low	high	low	high
1. Applicant=good	-	×	×	×	-	×	-	×
2. Applicant=bad	×	-	-	-	×	-	×	-

(a) Expanded DT

1. Owns property?	yes			no	
2. Years client	≤ 3		>3	-	
3. Savings amount	low	high	-	low	high
1. Applicant=good	-	×	×	-	×
2. Applicant=bad	×	-	-	×	-

(b) Contracted DT

1. Savings amount	low			high
2. Owns property?	yes		no	-
3. Years client	≤ 3	>3	-	-
1. Applicant=good	-	×	-	×
2. Applicant=bad	×	-	×	-

(c) Minimum-size contracted DT

Figure 4.11: Minimizing the number of columns of a lexicographically ordered DT [248].

1. Savings amount	high	-	low	low
2. Owns property?	-	yes	no	-
3. Years client	-	> 3	-	≤ 3
1. Applicant=good	×	×	-	-
2. Applicant=bad	-	-	×	×

Figure 4.12: Example of an unordered DT.

Note that we deliberately restrict ourselves to single-hit tables, wherein columns have to be mutually exclusive, because of their advantages with respect to verification and validation [247]. It is this type of DT that can be easily checked for potential anomalies, such as inconsistencies (a particular case being assigned to more than one class) or incompleteness (no class assigned). The decision table formalism thus facilitates the verification of the knowledge extracted by e.g. a neural network rule extraction algorithm. What's more, inspecting and validating a DT in a top-down manner, as suggested above, should prove more intuitive, faster, and less prone to human error, than evaluating a set of rules one by one. Then, in a final stage, once the decision table has been approved by the expert, it can be incorporated into a deployable decision support system [248].

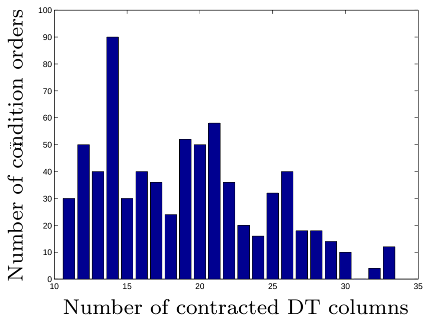
We will use the PROLOGA[4] software to construct the decision tables for the rules extracted in section 4.4.3. PROLOGA is an interactive design tool for computer-supported construction and manipulation of DTs [245]. With PROLOGA, knowledge is acquired and verified in the form of a system of DTs. A powerful rule language is available to help specify the DTs, and automated support is provided for several restructuring and optimization tasks. Furthermore, to assist in the implementation and integration of the modeled knowledge into various types of application settings, a range of import / export interfaces is included (e.g., code generation utilities), as well as a standard consultation environment, which allows the user to apply the knowledge to a given problem case by means of a targeted question / answer dialog, similar to that offered in a typical rule-based KBS shell.

Table 4.10 presents the properties of the DTs built for the rules extracted by Neurorule and Trepan on all three credit scoring data sets. For the German credit data set, the fully expanded decision table contained 6600 columns, enumerating every possible combination of distinct attribute values ($= 4 \times 5 \times 2 \times 11 \times 5 \times 3$). For the Bene1 and Bene2 data sets, the expanded table consisted of 192 columns. Though hardly suitable for visual inspection at this point, the expanded DTs proved computationally tractable given the input space reduction achieved in the preceding stages of the knowledge discovery process. Subsequently, we converted each of these expanded DTs into a more compact DT, by joining nominal attribute values that do not appear in any rule antecedent into a common 'other' state, and then performing optimal table contraction. Considering the limited number of inputs, we adopted a simple exhaustive search method (requiring only a few seconds on a Pentium 4); a branch-and-bound approach to find the optimal condition order is described elsewhere [246]. For the rules extracted by Neurorule, we ended up with three minimum-size contracted DTs, consisting of 11, 14 and 26 columns for the German credit, Bene1 and Bene2 data sets, respectively. For the Trepan trees, the minimized DTs had 9, 30 and 49 columns for the three data sets (cf. Table 4.10). Note that we converted the Trepan trees to an equivalent set of rules in order to make the decision tables. Since Nefclass gave rather bad performance on all data sets, we did not include decision tables for the extracted fuzzy rules. Remark however that decision tables can be easily adopted to model a

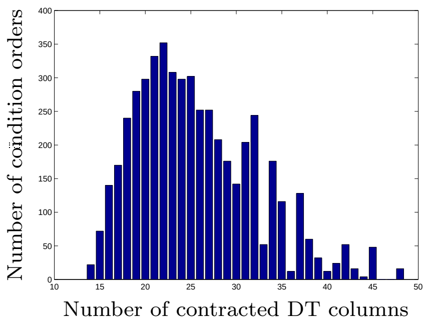
[4]`http://www.econ.kuleuven.ac.be/tew/academic/infosys/research/Prologa.htm`

set of descriptive fuzzy rules (fuzzy decision tables) [249, 260]. We also constructed the decision tables for the rules induced by C4.5rules and found that these tables were huge and impossible to handle because of the large number of generated rules and unpruned inputs.

Interestingly, the size gains achieved by the DT contraction mechanism were, in all cases, substantial, even with non-optimal condition orders. For example, for the Bene1 credit data set, the maximum (average) contracted DT size amounted to 48 (26), respectively, which is still well below the theoretical worst-case of 192 (non-contractable) columns. Figure 4.13 shows a bar plot of which the Y-axis depicts the number of condition orders leading to the DT size indicated on the X-axis for the German credit and Bene1 data sets.



(a) German credit



(b) Bene1

Figure 4.13: DT size distribution [9].

Figures 4.14 and 4.15 depict the contracted decision tables generated from the rules extracted by Neurorule for the German credit and Bene1 data sets. Clearly, their relative conciseness, combined with their top-down readability, is what makes them a very attractive visual representation of the extracted knowledge. It is important to note here that transforming a set of propositional rules into a DT does not entail any loss of predictive accuracy; i.e., the decision tables depicted in Figures 4.14 and 4.15 have exactly the same classification accuracy as the rules of Figures 4.6 and 4.7 from which they were generated. Hence, as no anomalies are indicated in the DT, the completeness and consistency of the extracted rules are demonstrated.

Decision tables allow for an easy and user-friendly consultation in every day business practice. Figure 4.16 presents an example of a consultation session in Prologa. Suppose we try to work ourselves towards column 12 of the decision table for Bene1 depicted in Figure 4.15. We start with providing the system with the following inputs: Savings account $\leq$ 12.40 Euro, Economical sector = other and Purpose = second hand car. At this point, the Term input becomes irrelevant (indicated by '-') and hence, the system prompts for the next relevant input which is the number of years the applicant has been a client of the bank. We then indicate that the applicant has been a client for more than 3 years. The other remaining inputs (Owns Property and Income) then become irrelevant which allows the system to draw a conclusion: Applicant=good. This is illustrated in Figure 4.17. For this particular applicant, the system needed only 4 of the 7 inputs to make a classification decision. This example clearly illustrates that the use of decision tables allows to ask targeted questions by neglecting the irrelevant inputs during the decision process. It is precisely this property that makes decision tables interesting tools for decision support in credit scoring.

1. Checking account	1 or 2										3 or 4
2. Credit History	0 or 1				2 or 3					4	-
3. Term	1			2	1	2				-	-
4. Purpose	1 or 5	8	other	-	-	1 or 5	8 or other			-	-
5. Savings account	-	-	-	-	-	-	1 or 2 or 3		4 or 5	-	-
6. Other parties	-	-	-	-	-	-	1 or 2	3	-	-	-
1. Applicant=good	-	×	-	-	×	×	-	×	×	×	×
2. Applicant=bad	×	-	×	×	-	-	×	-	-	-	-
	1	2	3	4	5	6	7	8	9	10	11

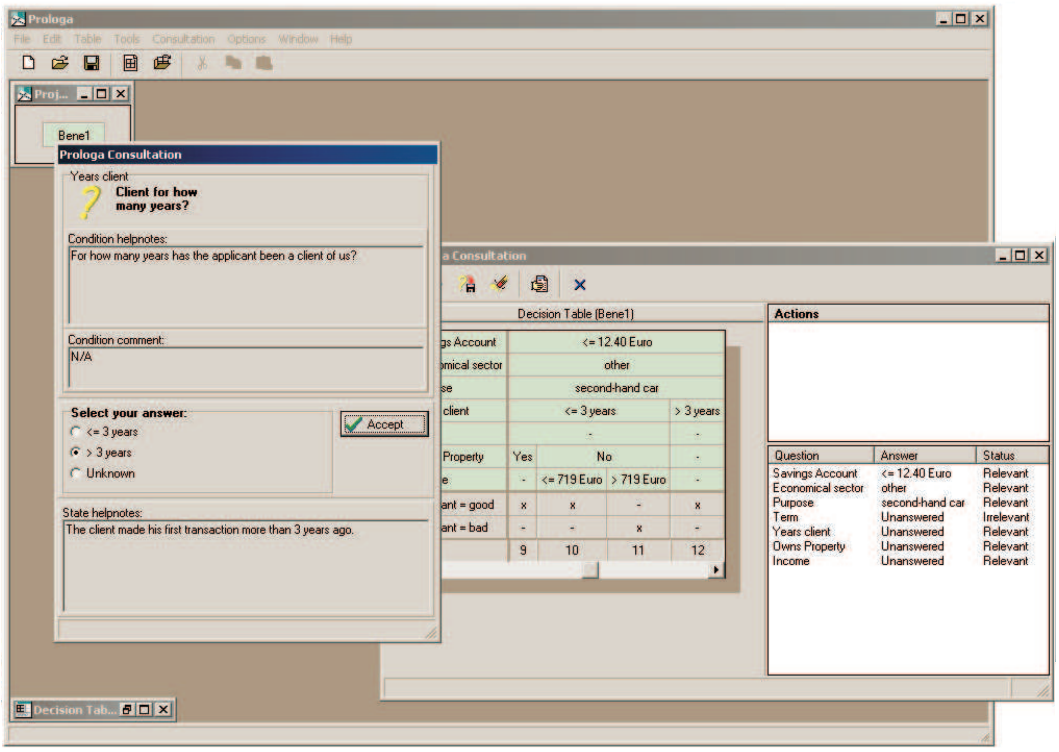
Figure 4.14: Decision table for the rules extracted by Neurorule on German credit.

1. Savings Account	≤12.40 Euro													> 12.40 Euro
2. Economical sector	Sector C	other												-
3. Purpose	-	cash provisioning							second-hand car				other	-
4. Term	-	≤ 12 months				> 12 months			-					
5. Years Client	-	≤ 3			>3	≤ 3	>3		≤ 3			> 3		
6. Owns Property	-	Yes	No		-	-	Yes	No	Yes	No		-	-	-
7. Income	-	-	≤ 719 Euro	> 719 Euro	-	-	-	-	-	≤ 719 Euro	> 719 Euro	-	-	-
1. Applicant=good	-	×	×	-	×	-	×	-	×	×	-	×	×	×
2. Applicant=bad	×	-	-	×	-	×	-	×	-	-	×	-	-	-
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Figure 4.15: Decision table for the rules extracted by Neurorule on Bene1.

Data set	**Extraction method**	**Number of columns in expanded DT**	**Number of columns in minimized DT**
German credit	Neurorule	6600	11
	Trepan	6600	9
Bene1	Neurorule	192	14
	Trepan	192	30
Bene2	Neurorule	192	26
	Trepan	192	49

Table 4.10: The number of columns in the expanded and reduced DTs for the three data sets for the rules and trees extracted by Neurorule and Trepan.



Prologa Consultation

Years client
Client for how many years?

Condition helpnotes:
For how many years has the applicant been a client of us?

Condition comment:
N/A

Select your answer:
<= 3 years [✓ > 3 years] Unknown

State helpnotes:
The client made his first transaction more than 3 years ago.

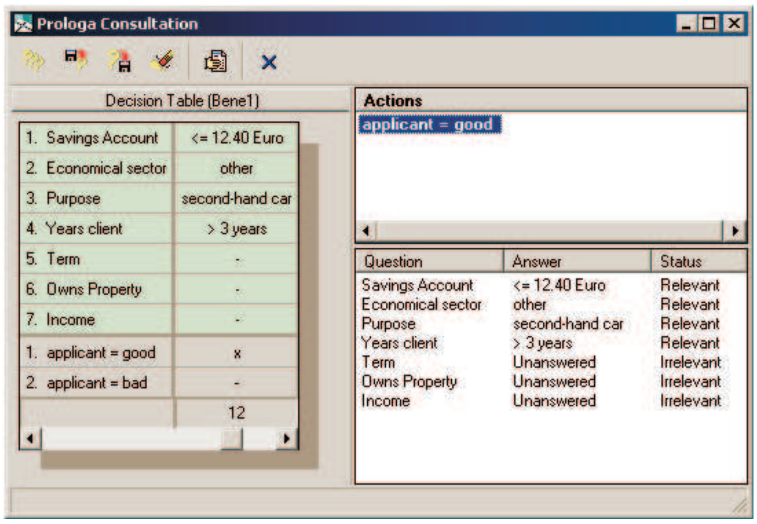
Decision Table (Bene1)

gs Account	<= 12.40 Euro			
omical sector	other			
se	second-hand car			
client	<= 3 years			> 3 years
	-			-
Property	Yes	No		-
e	-	<= 719 Euro	> 719 Euro	-
ant = good	x	x	-	x
ant = bad	-	-	x	-
	9	10	11	12

Actions

Question	Answer	Status
Savings Account	<= 12.40 Euro	Relevant
Economical sector	other	Relevant
Purpose	second-hand car	Relevant
Term	Unanswered	Irrelevant
Years client	Unanswered	Relevant
Owns Property	Unanswered	Relevant
Income	Unanswered	Relevant

Figure 4.16: Example consultation session in PROLOGA.



Prologa Consultation

Decision Table (Bene1)

1. Savings Account	<= 12.40 Euro
2. Economical sector	other
3. Purpose	second-hand car
4. Years client	> 3 years
5. Term	-
6. Owns Property	-
7. Income	-
1. applicant = good	x
2. applicant = bad	-
	12

Actions

applicant = good

Question	Answer	Status
Savings Account	<= 12.40 Euro	Relevant
Economical sector	other	Relevant
Purpose	second-hand car	Relevant
Years client	> 3 years	Relevant
Term	Unanswered	Irrelevant
Owns Property	Unanswered	Irrelevant
Income	Unanswered	Irrelevant

Figure 4.17: Classifying an applicant in Prologa.

4.6 Conclusions

Justifying and clarifying the decisions made by a credit scoring system is becoming more and more a key success factor for its successful deployment and integration into the daily credit decision environment. In the previous chapter, we have seen that neural networks are amongst the best techniques for building scorecards. However, their complex internal mathematical workings essentially turn them into black box models which provide no humanly comprehensible explanation of the classifications being made. In this chapter, we have shown how neural network rule extraction techniques try to solve this problem by opening the neural network black box and extracting a set of rules or trees which explain the behavior of the trained network. This will provide the credit scoring expert with an explanation facility to clarify why credit is either granted or denied. We believe neural network rule extraction is a promising and interesting approach to make credit scoring systems intelligent and increase their chances of being successfully deployed and effectively used.

In this chapter, we have investigated the following neural network rule extraction techniques: Neurolinear, Neurorule, Trepan and Nefclass. Neurolinear and Neurorule are both decompositional rule extraction algorithms whereas Trepan is a pedagogical tree extraction algorithm. Neurolinear works with continuous data and extracts oblique decision rules. On the other hand, Neurorule assumes binarized data and infers propositional rules. Trepan is a tree extraction algorithm extracting decision trees with M-of-N type of splits. We have also included Nefclass as an example of a neurofuzzy classifier that infers descriptive fuzzy rules.

The experiments were conducted on the continuous and the discretized versions of the German credit, Bene1 and Bene2 data sets. We have evaluated both the classification accuracy as measured by the percentage correctly classified observations, and the complexity of the extraction techniques. We believe the latter criterion is very important since simple, parsimonious rule sets or trees are easier to comprehend and thus easier to deploy and use in everyday business practice. We have compared the performance of the extraction algorithms with the performance of the well-known C4.5 and C4.5rules algorithms. It was found that, for the continuous data sets, Neurolinear achieved a very good test set performance with only a small number of oblique rules. However, the comprehensive value of these rules is rather limited since they represent piece-wise mathematical discriminant functions. For the discretized data sets, it was concluded that both Neurorule and Trepan yielded a very good performance in terms of classification accuracy and complexity. The propositional rules inferred by Neurorule are very comprehensible whereas the Trepan trees are less easy to understand due to the M-of-N type of splits. Nefclass never achieved a satisfying performance for both the continuous and the discretized data sets.

Recent research in knowledge representation suggests that graphical representation formalisms can be more readily interpreted and consulted by humans than a set of symbolic propositional if-then rules. Hence, in a next step, we investigated how the rules and trees inferred by Neurorule and Trepan can be represented using decision tables. Decision tables provide a tabular representation of the extracted rules and trees. They have specialized contraction and minimization mechanisms which allow for a powerful and parsimonious representation of the knowledge. We showed that the contracted and minimized decision tables built for the rules and trees extracted by Neurorule and Trepan on all 3 data sets were satisfactorily concise and did not contain any anomalies. Furthermore, it was also demonstrated that the use of decision tables allows one to ask targeted questions during the consultation process by neglecting the irrelevant inputs. Hence, using decision tables to represent extracted knowledge is an interesting and powerful alternative for building user-friendly, intelligent credit scoring systems.

The rules extracted in this chapter are crisp in the sense that their antecedent is either true or false. Fuzzy rules allow for more flexibility and express the antecedents in terms of linguistic concepts modeled by fuzzy sets. In the following chapter, we will investigate the use of genetic and neurofuzzy algorithms for fuzzy rule extraction.

Chapter 5

Building Intelligent Systems for Credit Scoring using Fuzzy Rule Extraction

In this chapter, we study the use of fuzzy rule extraction techniques for credit scoring[1,2]*. In contrast to crisp rules, fuzzy rules are often described as being more close to human reasoning since they are usually expressed in terms of linguistic concepts. We will study two types of fuzzy rules: descriptive fuzzy rules and approximate fuzzy rules. Descriptive fuzzy rules all share a common, linguistically interpretable definition of membership functions whereas approximate fuzzy rules each have their own definition of membership functions. Many learning paradigms have been suggested to extract fuzzy rules. We present two evolutionary fuzzy rule learners, an evolution strategy that generates approximate fuzzy rules and a genetic algorithm that extracts descriptive fuzzy rules. The performance of the evolutionary fuzzy rule learners is compared with that of Nefclass, a neurofuzzy classifier, and a selection of other well-known classification algorithms on a number of data sets including the Australian credit, German credit, Bene1 and Bene2 credit scoring data sets.*

[1]F. Hoffmann, B. Baesens, J. Martens, F. Put, J. Vanthienen, Comparing a Genetic Fuzzy and a Neurofuzzy Classifier for Credit Scoring, *International Journal of Intelligent Systems*, Volume 17(11), pp. 1067-1083, 2002.

[2]F. Hoffmann, B. Baesens, J. Martens, F. Put, J. Vanthienen, Evolutionary Algorithms for Inferring Descriptive and Approximate Fuzzy Classification Rules, Submitted for Publication.

5.1 Fuzzy Classification Rules

In chapter 4, we studied the use of crisp rule extraction techniques for credit scoring. A key property of a crisp rule is that each condition of the rule antecedent essentially models a crisp set. An example of a crisp set is:

$$A_j = \{x_j \,|x_j \;> 30\}, \tag{5.1}$$

for each value of the variable x_j whereby 30 is the cut-off point to decide if the element belongs to the set. In contrast to crisp rules, fuzzy rules allow for more flexibility by using fuzzy sets to express the conditions in the rule antecedent [265, 266]. A fuzzy set has no crisp boundary such that an element always belongs to it up to some degree. The degree of belonging is then quantified by the membership function of the fuzzy set. More formally, a fuzzy set A_j in x_j is defined as a set of ordered pairs:

$$A_j = \{(x_j, \mu_A(x_j))\}, \tag{5.2}$$

for each value of x_j. $\mu_A(x_j)$ is called the membership function of x_j in A_j. The membership function maps each value of the variable x_j to a continuous membership value between 0 and 1. Common types of membership functions are triangular, trapezoidal and Gaussian membership functions. Analogous to the classical set operations of union, intersection and complement, Zadeh defined the fuzzy set variants hereof in his seminal paper [265]. One of the advantages of using fuzzy set theory is that it allows to model vague, linguistic concepts, which play an important role in human thinking, in a natural way.

Example 5.1
Consider the linguistic concepts young, middle aged and old. Using crisp sets, one could state that a person is young if his age is lower than 18, middle aged if his age is between 18 and 70 and old if his age is higher than 70. However, it is clear that the cut-points are rather artificially chosen. E.g. a person of 2 years and somebody of 17 years are both considered as young while there is a substantial age difference. Fuzzy sets allow each person to be young to a certain extent, middle-aged to a certain extent and old to a certain extent. Figure 5.1 provides an example of the use of trapezoidal membership functions for expressing the fuzzy sets young, middle aged and old.

An elemental fuzzy classification rule is of the form:

$$R_i : \textbf{If}\, x_1 \,\text{is}\, A_{1i} \;\textbf{And} \ldots\, x_n \,\text{is}\, A_{ni} \;\textbf{Then}\, \text{Class} = c, \tag{5.3}$$

whereby x_j denotes the j^{th} input variable, A_{ji} a fuzzy set associated to input variable x_j and c the class label of the rule. The elemental rule of 5.3 can now be generalized to a disjunctive normal form (DNF) fuzzy rule where each input variable can be related to several fuzzy sets which are combined using a disjunctive operator [156]. For three input variables each partitioned into 5 fuzzy sets a DNF fuzzy rule looks like

$$\textbf{If}\, x_1 \,\text{is}\, \{A_{12} \;\textbf{Or}\; A_{14}\,\} \;\textbf{And}\, x_2 \,\text{is}\, \{A_{25}\} \;\textbf{And}\, x_3 \,\text{is}\, \{A_{31} \;\textbf{Or}\; A_{32}\} \;\textbf{Then}\, \text{Class} = c \tag{5.4}$$

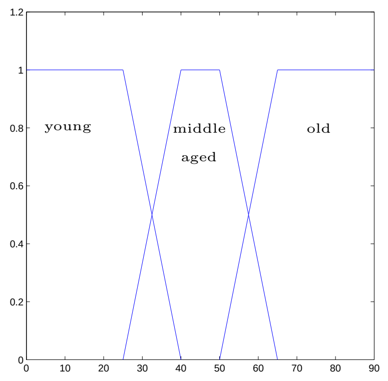


Figure 5.1: Fuzzy sets and corresponding membership functions for age.

One uses the t-norm to measure the rule activation for a particular instance **x**

$$\mu_{R_i}(\mathbf{x}) = \mu_{R_i}(\{x_1, ..., x_n\}) = \min_{j=1}^{n} \mu_{A_{ji}}(x_j). \tag{5.5}$$

Note that we hereby use a min operator instead of a product operator. The problem with the product operator is that for antecedents with many inputs, the degree of rule activation can become very low. The instance **x** is then classified in the class c_{max}

$$c_{max} = \mathrm{argmax}_{\mathrm{c_m}} \sum_{\mathrm{R_i|c_i=c_m}} \mu_{\mathrm{R_i}}(\mathbf{x}). \tag{5.6}$$

When the fuzzy sets A_{ij} are defined in exactly the same way in all fuzzy rules, the rules are called descriptive fuzzy rules. This facilitates the interpretation of the fuzzy sets as linguistic concepts such as old, hot, high, medium, ... Approximate fuzzy rules contain their own definition of membership functions rather than referring to a commonly defined set of membership functions. This means that the fuzzy sets A_{ij} can no longer be interpreted as linguistic concepts but instead refer to the characteristic functional relationship and corresponding parameters of the membership function. The difference between descriptive and approximate fuzzy rules is further clarified in Figure 5.2. It is obvious that approximate fuzzy rules are less comprehensible and user friendly than descriptive fuzzy rules.

5.2 Using Fuzzy Rules for Credit Scoring

Not many attempts have been made in the literature to use fuzzy rule extraction techniques for credit scoring.

Approximate fuzzy rules:

If Term is trapezoidal(19.2 31.9 70.2 81.4) **And** Property is trapezoidal(8.4 8.4 9.4 9.5)
Then Applicant=bad

If Term is trapezoidal(26.1 26.1 110.1 118.1) **And** Purpose is trapezoidal(2.3 2.9 5.2 6.2)
And Monthly Payment is trapezoidal (2502.2 6774.4 21242.9 21242.9)
Then Applicant=bad

If Purpose is trapezoidal(1.8 4.9 9.6 14.9) **And** Term is trapezoidal(24.3 24.5 108.5 108.5)
And Percentage of Financial Burden is trapezoidal(0.4 0.4 0.8 0.9)
Then Applicant= bad

Descriptive fuzzy rules:

If Percentage of Financial Burden is large **And** Code Regular Saver is large
Then Applicant=bad

If Percentage of Financial Burden is small
Then Applicant=good

If Percentage of Financial Burden is medium **And** Code Regular Saver is large
Then Applicant=bad

Figure 5.2: Approximate fuzzy rules versus Descriptive fuzzy rules [120]. Note that the approximate fuzzy rules are modeled using trapezoidal membership functions which are characterized by 4 coordinate points.

In [179], Piramuthu studied the use of neurofuzzy systems for financial credit-risk evaluation. The experiments were conducted using three real-life data sets: the Australian credit data set (653 Obs.), a data set with the loan repayment behavior of firms (48 Obs.) and two versions of a bankruptcy data set (162 Obs. and 158 Obs.). For each data set, the performance of the neurofuzzy classifier was compared with that of a neural network. The major conclusion was that neural networks yielded somewhat better classification accuracy than their neurofuzzy counterparts but the latter have the benefit of generating explanatory fuzzy rules, which is very important in a credit risk context.

Malhotra and Malhotra [157] used the ANFIS (adaptive network-based fuzzy inference system) method for credit scoring. ANFIS is a neurofuzzy-inference system described in [127]. They combined the data from nine credit unions to form a pooled data set with 790 observations with only three inputs: total payments to total income, total debt to total income and the credit rating of the applicant. To compare the performance of the classifiers, seven randomizations of the data were generated and each randomization was split, in a stratified manner, into a training set (500 Obs.) and a test set (290 Obs.). The performance of the neurofuzzy method was compared with linear discriminant analysis using paired t-tests. It was found that the neurofuzzy classifier statistically outperformed the discriminant analysis classifier in classifying the bad loans but not in classifying the good loans.

5.3 Evolutionary Algorithms for Inferring Descriptive and Approximate Fuzzy Rules

5.3.1 Evolutionary Algorithms

Evolutionary algorithms are general-purpose search algorithms based upon the principles of Darwinian evolution observed in nature [36, 38]. An evolutionary algorithm typically starts from a randomly initialized population of individuals (also called chromosomes), each representing a potential solution to the problem at hand, and tries to mimic the well-known principle of survival of the fittest by generating new, better populations using certain types of evolutionary operators. The algorithm first makes a selection of the best individuals according to a fitness function which reflects the quality of the solutions and which is dependent upon the problem statement. In a next step, the individuals can be mutated or recombined with other individuals. Mutation introduces new information into the population whereas recombination allows for exchanging information between individuals. This process is then iterated until a pre-specified fitness values is obtained or another stopping condition is met. The rough outline of a general evolutionary algorithm is given in Figure 5.3. Here t denotes the generation number and

```
t=0;
initialize(P(t));
evaluate(P(t));
while not terminate (P(t)) do
 t=t+1;
 P(t)=select(P(t-1));
 recombine(P(t));
 mutate(P(t));
 evaluate(P(t));
end
```

Figure 5.3: Outline of an evolutionary algorithm [36, 38].

$P(t)$ is the population at generation t. Two types of evolutionary algorithms can be distinguished. A genetic algorithm uses a binary representation of the individuals whereas evolution strategies consider vectors of real values to represent the individuals. Evolutionary algorithms are particularly well suited to deal with problems where no other specific techniques are available, e.g. multimodal, noisy objective functions, constraint models, simulation models, ... Their major advantage is that they have no assumptions with respect to the problem space. Major drawbacks are that there is no guarantee to find an optimal solution within a finite amount of time and that the parameters are often tuned in an ad-hoc way. For more details on evolutionary algorithms, the reader is referred to, e.g., [36, 37, 38, 97].

5.3.2 Boosted Genetic Fuzzy Classification

Many learning paradigms have been suggested to learn fuzzy classification rules. Amongst them, the use of evolutionary algorithms is becoming very popular. A genetic fuzzy rule based system (GFRBS) uses evolutionary algorithms to adapt an entire fuzzy rule base or its individual components [48]. Although most of the work on GFRBS is concentrated in the area of fuzzy control, the focus has recently shifted towards the generation of fuzzy classification rules [24, 47, 119, 121, 125]. The iterative rule learning (IRL) approach to GFRBS grows the fuzzy rule base in an incremental way [100]. An evolutionary algorithm extracts the fuzzy rules one by one from a set of training observations hereby removing all observations that match the fuzzy rule. This process is then repeatedly invoked until a stopping criterion is met. A post-processing stage can then further prune and refine the rule base in order to augment its predictive power.

Figure 5.4 depicts the architecture of the boosted evolutionary classifier that will be used in this chapter. The classifier combines the ideas of boosting and iterative fuzzy rule learning into a coherent framework. After having generated a fuzzy rule, the classifier invokes a boosting scheme which changes the training set distribution. This scheme essentially reweights the training set observations by assigning higher weights to the instances that are currently misclassified or uncovered by the newly generated fuzzy rule. In this way, a bias is created towards the generation of fuzzy rules that try to complement the current set of fuzzy rules and correct their deficiencies. It is precisely this bias that will make the post-processing faze redundant. All extracted fuzzy rules are then aggregated into a single, composite classifier [91]. The boosting mechanism works especially well when the rule generation algorithm is an unstable classifier, i.e. a classifier which is particularly sensitive to changes in the training set distribution. Boosting algorithms for fuzzy classification systems have been previously proposed in [118, 132]. The architecture of Figure 5.4 will be used to generate both the approximate and the descriptive fuzzy rules. However, both will use another genetic representation and evolutionary algorithm, as will be described in the following subsection.

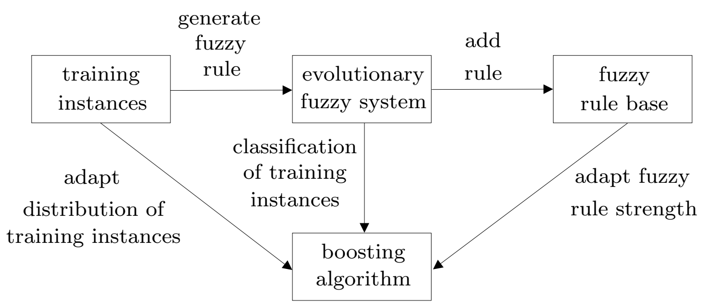

training instances
generate fuzzy rule
evolutionary fuzzy system
add rule
fuzzy rule base
adapt distribution of training instances
classification of training instances
adapt fuzzy rule strength
boosting algorithm

Figure 5.4: Architecture of a boosted evolutionary classifier.

5.3.3 Genetic Representation for Approximate and Descriptive Fuzzy Rules

A genetic algorithm is used to generate the descriptive DNF fuzzy rules. The latter are encoded using bit strings, in which each bit denotes the presence or absence of a linguistic term A_{ij} in the rule antecedent [156]. The rule in equation 5.4 would be encoded by the bitstring 01010|00001|11000 where the 1s correspond to the $A_{12}, A_{14}, A_{25}, A_{31}, A_{32}$. The number of fuzzy rules required grows rapidly with the number of input variables. In order to avoid an explosion of the number rules, the coding scheme provides for wildcard variables that are omitted from the rule antecedent. The chromosome contains an additional bit-string $S = \{s_1, \ldots, s_n\}$ in which the bit s_i indicates the presence or absence of the input variable x_i in the rule antecedent irrespective of the bits referring to the linguistic labels. Adaptation of the bit-string S enables the genetic algorithm to identify fuzzy rules with those inputs that best discriminate among the different classes, irrespective of the bits referring to the linguistic labels. Each chromosome also contains an additional bit representing the class label of the rule. The antecedent part of the chromosomes in the first generation is initialized by randomly picking a training instance. The bits corresponding to labels that best match the instance are set to 1, neighboring bits are chosen randomly and the remaining bits are set to 0. The initial rule class is determined by the majority class of those training instances covered by the rule antecedent. The input domains are partitioned into five respectively seven equally distributed triangular fuzzy sets which degrees of membership sum up to one.

Approximate fuzzy rules each have their own definition of membership functions. The fuzzy sets A_i thus no longer refer to linguistic concepts but instead typically vary from rule to rule. Equation 5.3 presents an example of the type of approximate rules that will be generated. Note that the approximate rules are less expressive than their descriptive counterparts since the latter allow for disjunctions in the antecedent part. The chromosomes are now real-valued vectors representing the characteristic points of the trapezoidal membership functions: the left most characteristic point a_i and the distances between the remaining points $\delta_i^1 = b_i - a_i$,

$\delta_i^2 = c_i - b_i$ and $\delta_i^3 = d_i - c_i$ (see Figure 5.5). The δ_i^j are restricted to positive values only such that the points a_i, b_i, c_i, d_i do not change their order. The entire rule chromosome then concatenates the characteristic points of the fuzzy sets $A_1, \ldots, A_n$ into one real-valued vector $a_1, \delta_1^1, \delta_1^2, \delta_1^3, \ldots, a_n, \delta_n^1, \delta_n^2, \delta_n^3$. Again the bit-string $S = \{s_1, \ldots, s_n\}$ is added to the chromosome to allow the algorithm to perform input selection. Due to the continuous nature of the optimization problem, an evolution strategy rather than a genetic algorithm is used for optimizing the fuzzy classification rules.

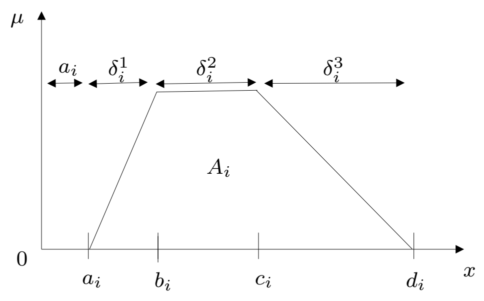


Figure 5.5: Coding of trapezoidal fuzzy sets.

5.3.4 Fitness Function

We will use the same fitness function to evaluate both the descriptive and approximate fuzzy rules. In [99], the authors propose the following optimization criteria to evaluate the quality of a fuzzy rule:

- class coverage;
- rule coverage;
- k-consistency.

It is important to note that each training example $\mathbf{x}_i$ has a corresponding weight w_i reflecting the frequency of the example in the training set. These weights will be modified by the boosting algorithm described in the next subsection. The fitness function will take into account these weights such that instances with large weights contribute more to the fitness than instances with small weights.

The first component of the fitness function is the class coverage defined as the ratio between the number of training instances covered by the rule R_i and the overall number of training instances carrying the same class label c_i,

$$f_1 = \frac{\sum_{k|c_k=c_i} w_k \mu_{R_i}(\mathbf{x}_k)}{\sum_{k|c_k=c_i} w_k}. \tag{5.7}$$

The second criterion is the rule coverage and is related to the overall fraction of instances covered by the rule, irrespective of the class label,

$$n = \frac{\sum_{k|c_k=c_i} w_k \mu_{R_i}(\mathbf{x}_k)}{\sum_k w_k}. \tag{5.8}$$

The idea is that a rule should cover a significant portion k_{cov} of the training examples instead of representing outliers in the training set

$$f_2 = \begin{cases} 1 & : n > k_{cov} \\ \frac{n}{k_{cov}} & \text{otherwise} \end{cases} \tag{5.9}$$

with $k_{cov} \approx 0.2 \ldots 0.5$ decreasing with an increasing number of classes. The reason is that with more classes it is harder to find rules that cover a significant fraction of all instances. For example, if you have a classification problem with M classes, each having the same number of instances, then a rule that perfectly classifies all instances of one class and none of the other $M-1$ classes has a coverage of $1/M$. A reasonable choice would then be $k_{cov} = 1/M$, as no rule can cover more than that fraction of instances without covering other (false) instances.

The number of correctly n_c^+ and incorrectly n_c^- classified weighted instances covered by the rule R_i is approximated as

$$\begin{aligned} {n_c}^+ &= \sum_{k|c_k=c_i} w_k \mu_{R_i}(\mathbf{x}_k) \\ {n_c}^- &= \sum_{k|c_k \neq c_i} w_k \mu_{R_i}(\mathbf{x}_k) \end{aligned} \tag{5.10}$$

Rule consistency demands that a rule covers a large number of correct instances and a small number of incorrect examples. Therefore, the fitness for the k-consistency of a rule is defined by the ratio of these numbers

$$f_3 = \begin{cases} 0 & : {n_c}^+ \times k < n_c^- \\ \frac{{n_c}^+ - n_c^-/k}{n_c^+} & : \text{otherwise} \end{cases} \tag{5.11}$$

where the parameter $k \in [0,1]$ determines the maximal tolerance for the error made by an individual rule [98]. The precise value of k depends on the problem complexity. Typically, k increases with an increasing overlap between classes. For binary classification problems a value of $k = 1$ was assumed.

The individual fitness criteria f_i are normalized to $[0, 1]$ and the overall fitness of a rule is given by the product

$$f = \prod_{i=1}^{3} f_i. \tag{5.12}$$

5.3.5 The Boosting Algorithm

The boosting scheme follows the idea of the iterative rule learning scheme and combines it with a fuzzy variant of the AdaBoost algorithm originally proposed in [91]. The basic idea of boosting is to repeatedly train a weak classifier on various distributions of the training data. After a fuzzy rule has been generated, one changes the distribution of training instances according to the observed error of the last generated classifier on the training set. The overall classification results from the aggregated votes of the individual classifiers.

If the learning algorithm can handle fractional training instances, it is not necessary to generate a new training set from a (modified) distribution. Instead, the instances obtain a weight w_k that specifies the relative importance of the k-th training instance. This weight can be interpreted as if the training set would contain w_k identical copies of the training example $(\mathbf{x}_k, c_k)$. Correctly classified instances $(\mathbf{x}_k, c_k)$ are down-weighted, such that the next iterations of the learning algorithm focuss on the seemingly more difficult instances.

Initially, all training examples obtain the weight $w_k = 1$. The boosting algorithm repeatedly invokes the genetic fuzzy rule generation method on the current distribution of training examples. Notice, that instances with large weights w_k contribute more strongly to the fitness of a fuzzy rule in equations 5.7, 5.9, and 5.11.

The boosting algorithm computes the error $E(R_t)$ of the fuzzy rule R_t generated in iteration t. Each descriptive or approximate fuzzy classification rule constitutes an incomplete, weak classifier. Incomplete in the sense that the rule only classifies instances covered by its antecedent but provides no classification for training examples that do not match the antecedent. Therefore, the classification error $E(R_t)$ of a fuzzy rule R_t is weighted not only by the weight w_k but also by the degree of matching $\mu_{R_t}(\mathbf{x}_k)$ between the k-th training instance $(\mathbf{x}_k, c_k)$ and the rule antecedent

$$E(R_t) = \frac{\sum_{k|c_k \neq c_t} w_k \mu_{R_t}(\mathbf{x}_k)}{\sum_k w_k \mu_{R_t}(\mathbf{x}_k)}. \tag{5.13}$$

In other words, the objective of the fuzzy rule generation method is to find classification rules that best describe the current distribution of training examples.

Assume that the rule generation algorithm identified R_t as the best rule for the distribution $w_k(t)$ at iteration t. For instances $(\mathbf{x}_k, c_k)$ correctly classified by

R_t the weight is reduced by a factor β_k, such that incorrectly classified instances gain relatively in importance in the next invocation of the genetic rule learner,

$$w_k(t+1) = \begin{cases} w_k(t) & \text{if } \mathrm{c_i} \neq \mathrm{c_k} \\ w_k(t) \times \beta_k^{\mu_{R_t}(\mathbf{x}_k)} & \text{if } \mathrm{c_i} = \mathrm{c_k}. \end{cases} \tag{5.14}$$

The factor

$$\beta_k = \frac{E(R_t)}{1 - E(R_t)} \tag{5.15}$$

depends on the error $E(R_t)$ of the fuzzy rule. Effectively, examples that are classified correctly and match the rule antecedent are down-weighted, and misclassified or uncovered examples keep their original weights. In other words, instances that are correctly classified get lower weights and the weight reduction will be higher when the rule activation is higher (for $\mu_{R_t}(\mathbf{x}_k) = 0$, the weight will remain unchanged). Thereby, the boosting algorithm increases the relative weight of those examples which are hard to learn for the genetic fuzzy system.

5.3.6 Fuzzy Classifier Aggregation

In order to classify unknown examples, the votes of the fuzzy rules R_t on a new instance $\mathbf{x}$ are aggregated into an overall classification [47, 124]. The classification proposed by a single rule is weighted according to the rule's classification accuracy expressed by

$$\beta_t = \frac{E(R_t)}{1 - E(R_t)}. \tag{5.16}$$

Rather than to pursue a winner-takes-all approach in which the rule with the highest matching degree dictates the ultimate classification, all rules contribute to the overall classification. The vote of rule R_t on $\mathbf{x}$ is weighted by the rule activation $\mu_{R_t}(\mathbf{x})$ and the factor $\log(1/\beta_t)$. The weight $\log(1/\beta_t)$ can be interpreted as the confidence in the fuzzy rule R_t. The boosting classifier then outputs the class label c_{max} that maximizes the sum

$$c_{max} = \mathrm{argmax}_{c_m} \sum_{R_t | c_t = c_m} \log(1/\beta_t)\, \mu_{R_t}(\mathbf{x}). \tag{5.17}$$

Intuitively, the most accurate and most active rules have the largest influence on the classification. Unfortunately, confidence rated fuzzy rules are less intuitive, in particular if several rules with possibly conflicting classes trigger for the same input. In that case, the outcome of the aggregated classification, not only depends on how well the rules match the instance, but also on their relative confidence.

5.4 Neural Network Fuzzy Rule Extraction using Nefclass

The category of neural network fuzzy rule extraction techniques is often referred to as neurofuzzy systems. Basically, these systems encompass methods that use learning algorithms from neural networks to tune the parameters of a fuzzy system. Although the interest in neurofuzzy systems originates mainly from the field of control theory, neurofuzzy methods may also be used for purposes of pattern classification or regression. Some of the most well-known neurofuzzy pattern recognition systems include FuNe [103], Fuzzy RuleNet [238], Anfis [127], Fuzzy Artmap [42] and Nefclass [170, 171, 172]. In this section, we will further elaborate on Nefclass since it is a neurofuzzy classifier that generates comprehensible descriptive fuzzy rules.

Nefclass has the architecture of a three-layer fuzzy perceptron whereby the first layer U_1 consists of input neurons, the second layer U_2 of hidden neurons and the third layer U_3 of output neurons. The difference with a classical multilayer perceptron (cf. section 2.1.9) is that the weights now represent fuzzy sets and that the activation functions are now fuzzy set operators. The hidden layer neurons represent the fuzzy rules and the output layer neurons the different classes of the classification problem with 1 output neuron per class. When a training pattern is propagated through the perceptron, the activation value of the hidden unit R and the output unit c are typically computed as follows:

$$\begin{aligned} a_R &= \min_{x_j \in U_1} \{W(x_j, R)(x_j)\}, \\ a_c &= \frac{\sum_{R \in U_2} W(R,c) a_R}{\sum_{R \in U_2} W(R,c)} \\ \text{or alternatively}: a_c &= \max_{R \in U_2} \{a_R\}, \end{aligned} \tag{5.18}$$

whereby $W(x_j, R)$ is the fuzzy weight between input unit x_j and hidden rule unit R and $W(R, c)$ is the weight between hidden rule unit R and output class unit c. Nefclass sets all weights $W(R, c)$ to 1 for semantical reasons. Depending on the classification problem at hand, the output activation function can be just a simple maximum-operator. After an observation has been propagated through the network, its predicted class is assigned according to the output neuron with the highest activation value (winner-takes-all). Figure 5.6 depicts an example of a Nefclass network. The fuzzy rule corresponding to rule unit R1 can then, e.g., be expressed as follows:

$$\textbf{If } x_1 \text{ is small } \textbf{And } x_2 \text{ is big } \textbf{Then } \text{Class} = C1, \tag{5.19}$$

where the fuzzy sets small and big have membership functions $\mu_1^{(1)}$ and $\mu_1^{(2)}$, respectively.

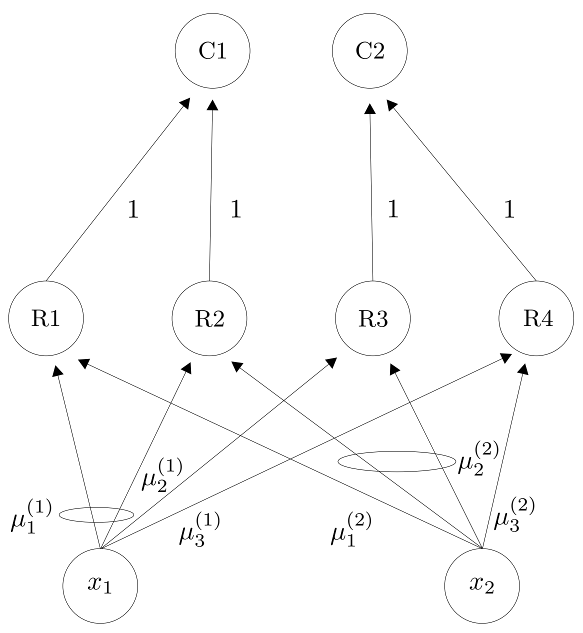


Figure 5.6: Example Nefclass network.

In order to generate descriptive fuzzy rules, rather than approximate fuzzy rules, Nefclass enforces all connections representing the same linguistic label (e.g x_1 is small) to have the same fuzzy set associated with them. E.g., in Figure 5.6, the fuzzy set having membership function $\mu_1^{(1)}$ is shared by the rule units R_1 and R_2 and thus has the same definition in both fuzzy rules. When this constraint was not imposed, the same linguistic label could be represented by different fuzzy sets which would lead to the generation of approximate fuzzy rules and thus decrease the interpretability and comprehensibility of the classifier.

Nefclass allows one to model a priori domain knowledge before starting to learn the various parameters or it can also be created from scratch. In both cases, the user must start by specifying the fuzzy sets and membership function types for all inputs which can be trapezoidal, triangular, Gaussian or List. The latter type of membership function is especially well-suited for dealing with nominal variables [170].

Nefclass starts by determining the appropriate number of rule units in the hidden layer. Suppose we have a data set D of N data points $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$,with input data $\mathbf{x}_i \in \mathbb{R}^n$ and target vectors $\mathbf{y}_i \in \{0,1\}^m$ for an m-class classification problem. For each input $x_j \in U_1$, q_j fuzzy sets $\mu_1^{(j)}, ..., \mu_{q_j}^{(j)}$ are defined. The rule learning algorithm then proceeds as follows.

1. Select the next pattern $(\mathbf{x}_i, \mathbf{y}_i)$ from D.
2. For each input unit $x_j \in U_1$, $j = 1, ..., n$ find the membership function $\mu_{l_j}^{(j)}$ such that
$$\mu_{l_j}^{(j)}(x_j) = \max_{l \in \{1,...,q_j\}} \{\mu_l^{(j)}(x_j)\}. \tag{5.20}$$
3. If there is no rule node R with:
$$W(x_1, R) = \mu_{l_1}^{(1)}, .., W(x_n, R) = \mu_{l_n}^{(n)} \tag{5.21}$$
then create such a node and connect it to output class node p if $\mathbf{y}_i(p) = 1$.
4. Go to step 1 until all patterns in D have been processed.

Obviously, the above procedure will result in a large number of hidden neurons and fuzzy rules. This can be remedied by specifying a maximum number of hidden neurons and keeping only the first k rule units created (*Simple rule learning*). Alternatively, one could also keep the best k rules (*Best rule learning*) or the best $\lfloor \frac{k}{m} \rfloor$ rules for each class (*Best per Class rule learning*).

Once the number of hidden units has been determined, the fuzzy sets between the input and hidden layer are tuned to improve the classification accuracy of the network. Hereto, Nefclass employs a fuzzy variant of the well-known backpropagation algorithm to tune the characteristic parameters of the membership functions (see [170, 171, 172] for more details).

In a third step, Nefclass offers the possibility to prune the rule base by removing rules and variables based on a simple greedy algorithm which uses the following heuristics.

- pruning by correlation
  inputs that have only a low correlation with the output may be deleted
- pruning by classification frequency
  rules that are responsible for the classification of only a few cases may be removed from the rule base
- pruning by redundancy
  the linguistic term that yields the minimal degree of membership in an active rule in the least number of cases is deleted
- pruning by fuzziness
  all linguistic terms that correspond to the fuzzy set with the largest support are removed from all rules

The pruning strategies are executed in the above order. After each step, Nefclass retunes the fuzzy sets, and makes the pruning permanent if the new rule base yields a better accuracy. The goal of this pruning is to improve the comprehensibility of the created classifier.

5.5 Empirical Evaluation

5.5.1 Data sets and Experimental Setup

Table 5.1 represents the characteristics of the data sets that will be used to evaluate the different classifiers. The Breast cancer, Pima, Australian credit and German

	Data set size	Inputs		
		Total	Continuous	Nominal
Breast cancer	698	9	9	0
Pima	768	8	8	0
Australian credit	690	14	6	8
German credit	1000	20	7	13
Gauss	4000	2	2	0
Bene1	3123	27	13	14
Bene2	7190	28	18	10

Table 5.1: Characteristics of data sets.

credit data set are retrieved from the UCI repository (`http://kdd.ics.uci.edu/`). The Gauss data set is an artificial data set generated from two two-dimensional Gaussian distributions centered at $(0, 0)$ and $(2, 0)$ with covariance matrices I and $4I$. The Bene1 and Bene2 data sets are two real-life credit scoring data sets that have been obtained from major Benelux (Belgium, The Netherlands and Luxembourg) financial companies. All these data sets will be used to train the evolutionary and neurofuzzy classifiers and compare their classification accuracy with a selection of well-known classification algorithms such as Fisher discriminant analysis (Fisher), linear discriminant analysis (LDA), artificial neural networks (ANN) and C4.5 decision trees. The Fisher, LDA and ANN classifiers are implemented using the PRTools Matlab^{TM} toolbox[3]. In order to compute the classification accuracy for the Breast cancer, Pima, Australian credit, German credit, Gauss and Bene1 data sets, we will generate 10 randomizations and split each of them into a training set and test set. Each classifier will then be trained and evaluated 10 times and its accuracy will be averaged. The averaged performances can then be compared using paired t-tests (see 2.56). For the Bene2 data set, we will use a single training set/test set since it has a large number of observations. We will then use McNemar's test to test the performance differences (see 2.61). We will also use this data set to illustrate some of the extracted fuzzy rule sets.

For the descriptive genetic and neurofuzzy classifiers, we experimented with 5 and 7 fuzzy sets for the continuous attributes. The fuzzy sets are of triangular shape and uniformly partition the universe of discourse such that the membership

[3]`http://www.ph.tn.tudelft.nl/~bob/PRTOOLS.html`

degrees sum up to one. For the approximate and descriptive genetic classifiers, the boosting scheme invoked the evolutionary algorithm for rule generation 20 times, generating an equivalent number of rules. In general, the classification error for the test set converged after about 15 rules at which time usually each training example was at least covered by one fuzzy rule. The train and test set classification rates are reported as the average performance of the aggregated classifiers composed of 16 up to 20 rules. The evolutionary rule generation scheme evolved a population of 200 individuals over 50 generations of which the overall best solution was added to the aggregated classifier. The chromosome length depends on the number of attributes and in the descriptive case also on the number of fuzzy sets. The genetic algorithm used fitness proportionate selection and fitness scaling, whereas the evolution strategy operated with μ, λ selection in which the 20 best individuals served as parents for the next generation of offspring.

For the Nefclass classifier, we used triangular fuzzy sets, best per class rule learning and set the maximum number of fuzzy rules generated to 50. We report both an optimistic and a pessimistic accuracy estimate. The optimistic estimate classifies all unclassified observations into the majority class whereas the pessimistic estimate considers them as misclassified.

5.5.2 Results

Table 5.2 reports the classification accuracy of all classifiers on the 7 data sets. The best average test set classification accuracy is underlined and denoted in bold face for each data set. Test set performances that are not significantly different at the 5% level from the best performance with respect to a one-tailed paired t-test are tabulated in bold face. Statistically significant underperformances at the 1% level are emphasized. Performances significantly different at the 5% level but not a the 1% level are reported in normal script.

It can be observed from Table 5.2 that the Fisher, ANN, and the GFS Desc 7 FS classification techniques each achieved one time the best performance and the LDA and GFS Approx classification techniques each two times. Generally speaking, the GFS Desc 7 FS classifier performed only slightly better than the GFS Desc 5 FS classifier in absolute terms. The performance of the GFS Approx classifier was in line with that of its descriptive variants. Except for the Breast cancer and German credit data set, the GFS classifiers do not perform significantly worse than the standard classification schemes. The reason for the relative poor performance of the GFS classifiers on the Breast cancer and German credit data sets is related to the fitness criterion rule consistency in equation 5.11, for which throughout the experiments the default parameter value $k = 1$ was assumed. The parameter k should be adjusted to the difficulty of the classification problem, namely a small value of k if classes are easily separable as in the Breast Cancer data set, and a larger value of k if there is a significant overlap among classes as for example in the

Technique	Breast Cancer		Gauss		Pima		Australian credit		German credit		Bene1		Bene2	
	train	test	train	test	train	test	train	test	train	test	train	test	train	test
Fisher	96.6(0.4)	**95.9(0.9)**	76.1(0.7)	*76.1(0.5)*	77.8(0.9)	**77.3(0.8)**	86.0(0.8)	**86.7(1.5)**	77.1(0.8)	**76.3(1.6)**	72.9(0.2)	*71.3(0.9)*	73.9	**72.7**
LDA	96.6(0.4)	**95.8(0.9)**	76.1(0.7)	*76.1(0.5)*	77.9(1.1)	**77.3(0.7)**	86.0(0.8)	**86.7(1.6)**	77.5(0.8)	**76.0(1.4)**	72.7(0.5)	*71.2(0.8)*	74.4	**73.2**
ANN	97.9(0.8)	**96.2(0.8)**	78.9(1.5)	*78.4(1.4)*	77.4(3.0)	75.0(3.0)	90.5(1.2)	**86.0(2.1)**	81.9(1.3)	74.1(1.9)	75.9(1.0)	*71.5(1.0)*	73.5	**72.8**
C45	98.6(0.8)	*93.5(1.0)*	81.1(0.6)	*79.0(0.8)*	86.1(4.5)	*71.6(4.2)*	90.8(1.5)	**85.9(2.1)**	84.9(2.5)	*72.4(1.3)*	85.8(1.4)	*70.9(1.2)*	89.2	*70.2*
GFS Approx	94.9(1.9)	*92.7(2.1)*	81.1(0.6)	*79.4(0.8)*	84.0(0.8)	*75.9(1.6)*	87.1(1.4)	84.6(2.6)	80.4(1.0)	*73.3(2.2)*	74.6(0.4)	**72.9(1.2)**	75.2	**73.3**
GFS Desc 5 FS	91.0(0.6)	*89.3(0.8)*	79.6(0.6)	*79.6(0.5)*	80.6(1.3)	**76.0(1.9)**	88.9(0.9)	**85.9(2.2)**	80.3(1.0)	*73.2(2.0)*	73.6(0.4)	72.1(1.2)	75.1	**72.3**
GFS Desc 7 FS	93.5(0.6)	*91.0(1.2)*	80.4(0.4)	**80.3(0.4)**	81.4(1.9)	**76.6(1.7)**	89.4(1.1)	85.8(2.1)	80.2(0.8)	*73.4(1.6)*	73.8(0.6)	**72.9(0.9)**	74.8	**73.3**
Nefclass pess 5 FS	94.1(1.4)	*93.2(1.3)*	76.9(1.5)	*76.6(1.3)*	73.5(2.4)	*72.7(3.8)*	85.0(0.7)	**86.5(1.4)**	70.0(2.6)	*70.5(3.1)*	49.9(9.7)	*51.0(11)*	70.6	*70.1*
Nefclass opt 5 FS	94.2(1.4)	*93.3(1.4)*	77.6(1.9)	*77.3(1.8)*	74.6(0.8)	*73.8(2.2)*	85.0(0.7)	**86.5(1.4)**	70.2(2.6)	*70.6(3.3)*	67.3(7.7)	*68.6(2.1)*	71.2	*70.6*
Nefclass pess 7 FS	92.4(1.4)	*91.4(1.9)*	74.9(0.7)	*74.9(1.1)*	73.4(2.1)	*72.6(3.1)*	85.0(0.7)	**86.5(1.4)**	70.7(2.2)	*70.0(3.0)*	50.8(16)	*49.7(16)*	68.7	*69.4*
Nefclass opt 7 FS	92.4(1.3)	*91.5(1.9)*	75.6(0.5)	*75.3(1.4)*	74.9(1.1)	*74.2(1.7)*	85.0(0.7)	**86.5(1.4)**	71.5(1.5)	*70.7(2.3)*	69.1(3.9)	*68.6(3.6)*	69.4	*70.3*

Table 5.2: Classification accuracy of the evolutionary and neurofuzzy classifiers versus a selection of well-known classification algorithms.

credit data sets. For the Breast Cancer data set, a lower value of k would put a stronger emphasis on rule consistency relative to the weight given to the two other criteria rule and class coverage. Except for the Australian credit data set, both the C4.5 and Nefclass classifiers always achieved a statistically inferior classification performance at the 1% level. For the Nefclass classifier, we also experimented with other parameter settings (e.g. trapezoidal fuzzy sets, maximum 100 fuzzy rules, best rule learning) but found that its performance was highly dependent upon the specific parameter setting. Figure 5.7 shows the evolution of the training and

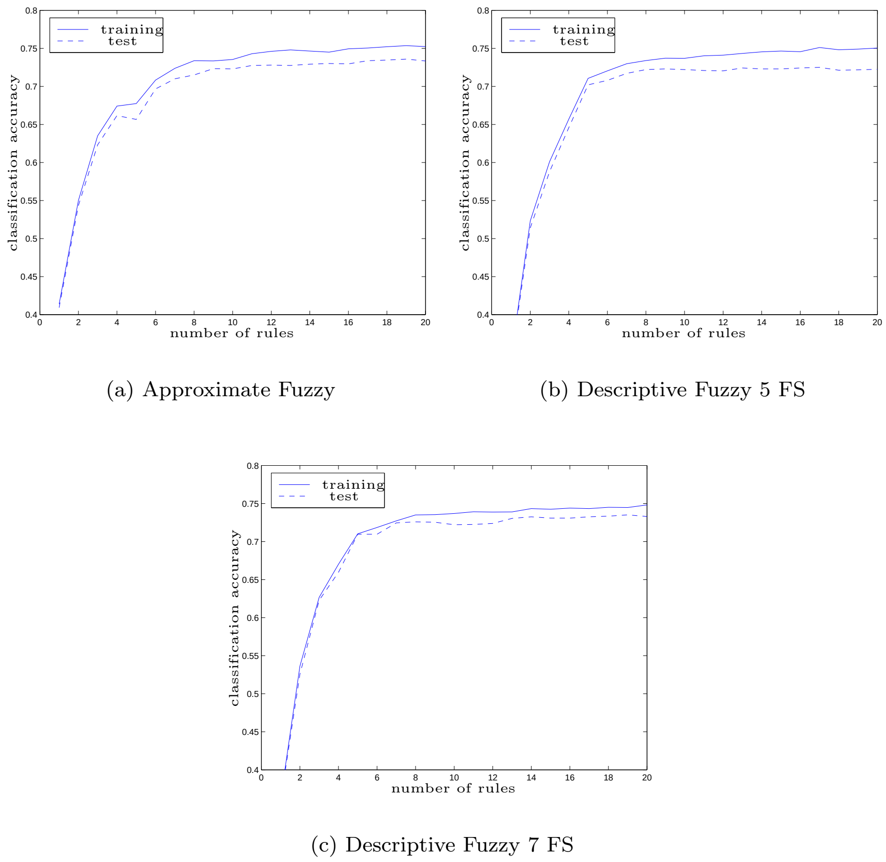


Figure 5.7: Evolution of the training and test set classification accuracy with the number of fuzzy rules generated for the evolutionary fuzzy classifiers on the Bene2 data set.

test set classification accuracy with the number of fuzzy rules generated for the evolutionary fuzzy classifiers on the Bene2 data set. The low classification rate for a small number of rules is due to the fact that a substantial number of instances is initially not covered by any rule and are therefore counted as misclassified. The figure clearly illustrates that for all three evolutionary classifiers, the classification accuracy on both the training set and test set stagnates after approximately 15 rules have been generated and no significant overfitting occurs when more fuzzy rules are added to the knowledge base.

One might expect that the approximate fuzzy classifiers perform better on data sets with dominantly continuous attributes and the descriptive version better on those with a larger number of nominal attributes. However, the results in Table 5.2 do not provide evidence to support this assumption. Even though single approximate and descriptive rules differ in their expressive power, it seems that the boosting scheme compensates for these limitations. The same applies to the resolution of rules, as the descriptive GFS with five fuzzy sets does not perform significantly worse than the GFS with seven fuzzy sets.

Figure 5.8 depicts the approximate fuzzy rules that were generated by the genetic fuzzy classifier for the Bene2 data set. Figure 5.9 presents the descriptive fuzzy rules extracted by the evolutionary fuzzy classifier using 5 fuzzy sets. One can observe, that for data sets with a large number of attributes, such as the Bene2 credit data, the approximate GFS scheme generates more compact rule antecedents that only refer to two or three attributes, whereas the descriptive rule antecedents utilize more attributes. The DNF descriptive rules are better able to cope with nominal attributes that do not obey an ordering relation, as they can form a disjunction of multiple allowed nominal values. On the other hand, approximate fuzzy rules offer an advantage for continuous attributes, in particular if the values are not uniformly distributed across the universe of discourse as assumed by the uniform descriptive fuzzy partitions. Also note how the weights complicate the interpretation of the fuzzy rules individually. E.g., some of the rules depicted in Figure 5.9 have very similar antecedents, but nevertheless propose opposite classifications. Which of these classifications ultimately dominates, can only be understood if one also takes the relative confidence levels of the rules into account.

If Term is trap(5 5.4 51.3 57.5) **And** Number of Years in Belgium is trap(14.5 28.1 73 83.6) **And** Age of Savings Account is trap(5.1 5.9 6 14.7) **Then** Customer=Good **w**= 1.70

If Number of Dependents is trap(0 0.7 6.2 7) **And** Parter Signed is trap(0.6 0.7 1.7 2.6) **Then** Customer=Good **w**= 0.67

If Percentage of Financial Burden is trap(14 22.8 63.1 63.2) **And** Purpose is trap(3.9 3.9 8.6 8.6) **Then** Customer=Bad **w**= 0.49

If Age of Savings Account is trap(4.4 4.7 5.7 6.3) **Then** Customer=Good **w**= 0.51

If Term is trap(24.2 36 63.9 65.3) **And** Property is trap(1.0 1.7 3.1 3.8) **Then** Customer=Bad **w**= 0.32

Figure 5.8: Approximate fuzzy rules for the Bene2 data set. The weights **w** correspond to the $\log(1/\beta_t)$ factors (see equation 5.17).

If Monthly Amount is very low **And** Stock Ownership is very low or low or high **And** Loan Amount is very low or medium or very high **And** Professional Income is very low or high **And** Number of Years in Belgium is very low **And** Purpose of Loan is 1 or 2 or 3 or 4 or 5 or 6
Then Customer = Good **w** = 1.76

If Monthly Amount is very low or high or very high **And** Amount on Savings Account is very low or low or medium or high **And** Non-professional Income is very low or low or medium **And** Number of Years at Current Address is very low or medium **And** Number of Years in Belgium is very low or low or very high **And** Total Income is very low or low or high or very high **And** Available Income is very low or low or medium or high or very high **And** Partner Signed is 0 or 1 **And** Marital Status is 1 or 2 or 3 or 4 or 5 **And** Works in Foreign Country is 0
Then Customer = Good **w** = 0.83

If Monthly Amount is very low or low **And** Amount on Savings Account is very low or low or medium or high **And** Percentage of Mortgage Burden is very low or very high **And** Other loan expenses is very low or medium or high **And** Non-professional Income is very low or medium **And** Professional Income is very low or very high **And** Number of Years at current Job is very low or high **And** Total Income is very low or very high **And** Available Income is very low or low or high or very high **And** Purpose of Loan is 1 or 2 or 5 **And** Profession is 1 or 3 or 4 or 6 or 13 or 15 or 16 or 18 or 21 or 27 or 28 or 29 **Then** Customer= Good **w** = 0.41

If Monthly Amount is very low or high **And** Amount on Savings Account is very low **And** Percentage of Mortgage Burden is very low or high or very high **And** Loan Amount is very low or low or medium or high or very high **And** Number of Years in Belgium is very low or low or high or very high **And** Total Income is very low or medium or high or very high **And** Available Income is very low or low or high or very high **And** Marital Status is 1 or 2 or 6 **Then** Customer=Bad **w** = 0.47

If Percentage of Mortgage Burden is very low or low **And** Other loan expenses is very low or medium **And** Loan Amount is very low **And** Professional Income is very low or high or very high **And** Number of Years at Current Address is very low or high **And** Available Income is very low or high **And** Partner Signed is 1 **And** Purpose of Loan is 1 or 2 or 3 or 4 or 6 **And** Loan Amount is very low **And** Profession is 1 or 4 or 5 or 9 or 13 or 14 or 15 or 16 or 20 or 26 or 28 or 29
Then Customer = Good **w** = 0.32

Figure 5.9: Descriptive fuzzy rules for the Bene2 data set using 5 fuzzy sets. The weights **w** correspond to the $\log(1/\beta_t)$ factors (see equation 5.17).

5.6 Conclusions

In this chapter, we proposed the use of fuzzy classification rules for credit scoring. Fuzzy rules are believed to be more comprehensible than crisp rules because they are expressed in terms of linguistic concepts which are more close to human reasoning. Many learning paradigms have been suggested to learn fuzzy classification rules. We studied the use of evolutionary algorithms and a neurofuzzy classifier. Two types of fuzzy rules were distinguished. Approximate fuzzy rules each have their own specific definition of membership functions, whereas descriptive fuzzy rules share a common, linguistically interpretable definition of membership functions in disjunctive normal form. It is obvious that the latter are easier to understand and comprehend than the former.

A genetic algorithm was used to infer the descriptive fuzzy rules whereas an evolution strategy was adopted to extract the approximate fuzzy rules. Both evolutionary algorithms used a boosting scheme that adapts the training set distribution such that the classifier focusses on currently mis- or unclassified observations. The performance of both evolutionary classifiers was compared with that of Nefclass, a neurofuzzy classifier generating descriptive fuzzy rules, and with a selection of well-known classification algorithms such as Fisher discriminant analysis, linear discriminant analysis, neural networks, and C4.5 decision trees. The experiments were carried out on a number of UCI data sets, amongst which the Australian credit and German credit data set, and the Bene1 and Bene2 data sets. It was shown that the evolutionary fuzzy rule learners compare favorably to the other classification techniques in terms of classification accuracy. Furthermore, the approximate and descriptive fuzzy rules, extracted by the evolutionary fuzzy rule learners, yield about the same classification accuracy across the different data sets. The boosting scheme seems to compensate for the expressive weaknesses of the individual classification rules. As a result no representation performs significantly better in terms of classification accuracy than the other. If the individual rules are more expressive, the same classification accuracy can be achieved with a smaller rulebase. The designer of a genetic fuzzy classifier faces a trade-off between a smaller number of more complex rules and a larger rulebase composed of more intuitive linguistic rules. Which design alternative is better, depends on the requirements of the application and whether compactness or interpretability of the rulebase are more important.

Although the descriptive fuzzy rules are more intuitively comprehensible than their approximate counterparts, it has to be noted that the weighting and corresponding voting scheme clouds the interpretation. Indeed, multiple rules typically contribute to the classification of an instance, such that the overall view of why it is classified into a specific class gets blurred. Hence, an important topic for further research might be to adapt the evolutionary algorithm to infer exhaustive, and mutually exclusive descriptive fuzzy rules which are not weighted and not aggregated using voting schemes.

Chapter 6

Survival Analysis for Credit Scoring

Traditionally, credit scoring aimed at distinguishing good payers from bad payers at the time of the loan application. However, the issue of when customers become bad is also very interesting to investigate since it can provide the bank with the ability to compute the profitability over a customer's lifetime and perform profit scoring. The problem statement of analysing when customers default is commonly referred to as survival analysis. Many survival analysis techniques have been suggested in a medical context. It is the purpose of this chapter to discuss and contrast statistical and neural network approaches for survival analysis in a credit scoring context[1]*. When compared to the traditional statistical proportional hazards model, neural networks may offer an interesting alternative because of their universal approximation property and the fact that no baseline hazard assumption is needed. Several neural network survival analysis models are discussed and evaluated according to their way of dealing with censored observations, time-varying inputs, the monotonicity of the generated survival curves and their scalability. In the experimental part of this chapter, we contrast the performance of a neural network survival analysis model with that of the well-known proportional hazards model for predicting both loan default and early repayment using data from a U.K. financial institution.*

[1]B. Baesens, T. Van Gestel, M. Stepanova, J. Vanthienen, Neural Network Survival Analysis for Personal Loan Data, Proceedings of the Eighth Conference on Credit Scoring and Credit Control (CSCCVII'2003), Edinburgh, Scotland, September, 2003.

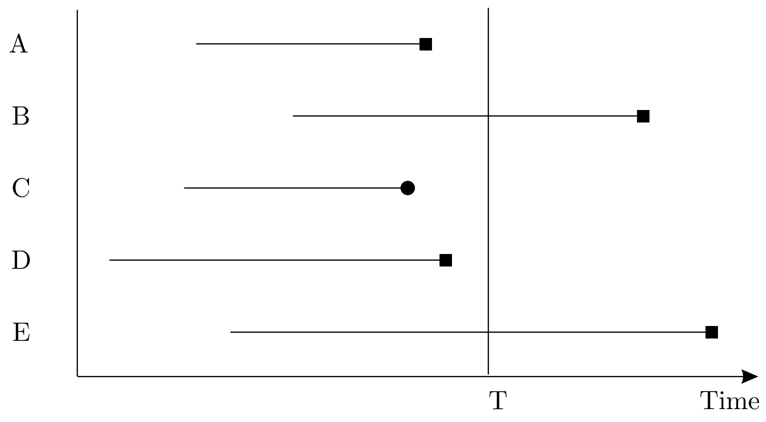


Figure 6.1: Censoring.

6.1 Basic Survival Analysis Concepts

Survival analysis deals with the problem of studying the occurrence and timing of events [3, 49, 133, 161]. The primary goal is to predict at what time an observation will undergo a particular event. Such an event is a qualitative change that can be situated in time, a transition from one discrete state to another [3]. Survival analysis then tries to predict the survival time of subjects by using descriptive properties called inputs or covariates. In today's literature, survival analysis is primarily applied to the study of deaths in a medical context. In [40, 58, 160, 182, 189, 223], for example, the influence of specific characteristics of patients with breast cancer on their survival time was studied. In [79, 268], survival analysis was applied to investigate the recurrence of prostate cancer carcinoma while in [174, 175] the same was done for AIDS. Other interesting application areas include credit scoring [21, 221] where survival analysis is used to predict default times, and marketing applications such as estimating interpurchase times [45, 115, 126] or customer lifetime [65, 159].

Two common features that are typical for survival data complicate the use of classical regression techniques: censoring and time-dependent variables [3, 49, 133, 161]. Ideally, each subject would be observed until the event occurs. However, some subjects may not have reached their event times when the study is terminated. These are called (right) censored observations. In a medical context, for example, patients may still be alive by the end of the study or may have died from causes not relevant to the study. Suppose Figure 6.1 represents data from a medical follow-up study investigating the survival of patients after heart surgery. The vertical line at time T indicates the end of the study. It is clear that patients A and D are not censored since they both die before T. However, the death times of patients B and E are not observed during the study and are thus censored at time T. Note that patients may also become censored during the time of the study. The death time of patient C is censored because e.g. he/she refuses to further participate in the study or has moved to another country and it is impossible to contact him/her. Note that thus far we have only considered right-censored observations where the time of the event is always bigger than the time of censoring. Left censoring occurs when the time of the event is smaller than some value but is less likely to occur in a survival analysis setting.

The second characteristic of survival data is the appearance of time-dependent inputs. The values of these inputs can vary over subsequent time periods, and are equally difficult to model using conventional regression techniques.

The aim of survival analysis is to estimate the distribution of the event times $f(t)$ of a group of subjects. This is usually done by specifying two other mathematically equivalent functions, the survival function $S(t)$ and the hazard function $h(t)$ defined as follows [3, 49, 133]:

$$S(t) = P(T > t) = \int_t^{\infty} f(u)du = 1 - F(t) \tag{6.1}$$

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t}, \tag{6.2}$$

whereby $F(t)$ represents the cumulative distribution function. The survival function models the probability that a subject will survive time period t. It is a monotone decreasing function of t with $S(0) = 1$ and $S(\infty) = 0$. Note that the integral is replaced by a summation for discrete distributions $f(t)$. Since time is continuous, the probability that an event will take place exactly at time t is 0. Hence, the hazard looks at the probability of an event occurring in time interval t to $t + \Delta$t, given that the subject has survived to t. However, this probability increases with Δt, and this is compensated by dividing by Δt. Taking the limit for $\Delta t \to 0$, the hazard function tries to quantify the instantaneous risk that an event will occur at time t given that the subject has survived to time t. It is a non-negative function of t and can be greater than 1.0 because of the division by Δt. It can also be thought of as the number of events per interval of time.

Since,

$$f(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t)}{\Delta t}, \tag{6.3}$$

and, by the definition of conditional probability, we have: $h(t) = f(t)/S(t)$. Since $f(t) = -dS(t)/dt$, we have $h(t) = -dlog(S(t))/dt$ and:

$$S(t) = \exp(-\int_0^t h(u)du). \tag{6.4}$$

The integral between brackets is called the cumulative hazard (or cumulative risk) and is denoted

$$H(t) = \int_0^t h(u)du. \tag{6.5}$$

It can be considered as the sum of the risks that are faced when going from time 0 to t. Using all these relationships, it becomes clear that once we know either $f(t)$, $h(t)$ or $S(t)$, the other two can be derived in a straightforward way.

6.2 Statistical Methods for Survival Analysis

6.2.1 Kaplan Meier Analysis

The Kaplan-Meier (KM) or product limit estimator is the non-parametric maximum likelihood estimator of the survival function $S(t)$ [134]. In the absence of censoring, the KM estimator $\hat{S}(t)$ is just the proportion of subjects having event times greater than t. When censoring is present, we start by ordering the event times in ascending order, $t_{(1)} < t_{(2)} < ... < t_{(k)}$[2]. The maximum likelihood KM estimator for the survival function then becomes [134]:

$$\hat{S}(t) = \prod_{j|t_{(j)}\leq t} (\frac{n_j - d_j}{n_j}) = \prod_{j|t_{(j)}\leq t} (1 - \frac{d_j}{n_j}) = \hat{S}(t-1)(1 - \frac{d_t}{n_t}) \qquad (6.6)$$

where d_j is the number of subjects with event time $t_{(j)}$ and n_j is the total number of subjects at risk at time $t_{(j)}$. A subject is at risk at time $t_{(j)}$ if its event or censoring time is greater than or equal to $t_{(j)}$. Note that in (6.6), $\hat{S}(t-1)$ represents the survival function estimate at the event time immediately preceding t.

6.2.2 Parametric Survival Analysis Models

Parametric survival analysis models approximate the lifetime distribution $f(t)$ by using popular distribution functions e.g. the exponential, Weibull and Gompertz distribution [3, 49, 133]. The distribution parameters are then estimated by maximizing the following likelihood function:

$$\prod_{i=1}^{N}[f(t_i)]^{\delta_i}[S(t_i)^{1-\delta_i}], \qquad (6.7)$$

where i runs over all event and censoring times and δ_i indicates if the observation is censored at t_i ($\delta_i = 0$) or not ($\delta_i = 1$). Indeed, if the subject fails at t_i ($\delta_i = 1$), its contribution to the likelihood function is $f(t_i)$ whereas if the individual is censored at t_i ($\delta_i = 0$) its contribution to the likelihood is the probability of surviving beyond t_i, $S(t_i)$.

[2]Note that these are the event times and not the censoring times and thus $k < N$.

Example 6.1
If, e.g., we assume that $f(t)$ follows an exponential distribution, we have $f(t) = \lambda \exp(-\lambda t)$, $S(t) = \exp(-\lambda t)$ and $h(t) = \lambda$. The likelihood function then becomes:

$$L = \prod_{i=1}^{N} [\lambda \exp(-\lambda t_i)]^{\delta_i} [\exp(-\lambda t_i)]^{1-\delta_i}. \tag{6.8}$$

Taking logarithms, we have

$$LL = \sum_{i=1}^{N} [\delta_i(\log \lambda - \lambda t_i) + (1-\delta_i)(-\lambda t_i)] = d \log \lambda - \lambda \sum_{i=1}^{N} t_i, \tag{6.9}$$

with d the number of subjects that fail: $d = \sum_{i=1}^{N} \delta_i$. Differentiating with respect to λ, we obtain:

$$\frac{\partial LL}{\partial \lambda} = \frac{d}{\lambda} - \sum_{i=1}^{N} t_i. \tag{6.10}$$

Equating this to zero yields

$$\hat{\lambda} = \frac{d}{\sum_{i=1}^{N} t_i}. \tag{6.11}$$

Figure 6.5 represents the survival and hazard functions for an exponentially distributed event time distribution $f(t)$ with $\lambda = 0.75$.

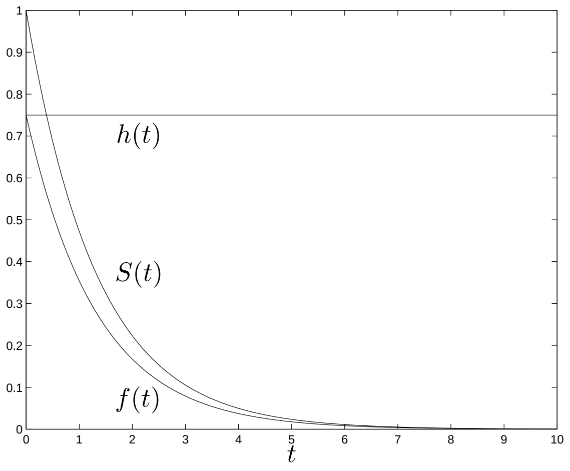


Figure 6.2: Survival and hazard functions for an exponentially distributed event time distribution $f(t)$ with $\lambda = 0.75$.

The disadvantage of these parametric survival analysis models is that they are not flexible enough to fit the data well because we have to assume a parametric function for the distribution $f(t)$. The smoothness of these popular distributions can be very restrictive, especially if the hazard function shows spikes because of specific events (e.g. contract expiration dates in a customer lifetime study [159]).

6.2.3 Proportional Hazards Models

The most common model used for survival analysis is the Cox proportional hazards model. The proportional hazards model (also called the Cox model) allows for the inclusion of explanatory inputs which may influence the survival time. It assumes that the inputs have a multiplicative effect on the hazard rate as follows [3, 49, 133]:

$$h(t, \mathbf{x}_i) = h_0(t) \exp[\boldsymbol{\beta}^T \mathbf{x}_i]. \tag{6.12}$$

This equation says that the hazard for a subject i at time t is the product of an unspecified, positive baseline hazard function $h_0(t)$, and a linear function of a vector of inputs $\mathbf{x}_i$ which is exponentiated. The baseline hazard $h_0(t)$ is a function of time only, and is assumed to be the same for all subjects. Since no intercept is included in the summation, we have that $h(t, \mathbf{x}_i) = h_0(t)$ for a subject i whose inputs $\mathbf{x}_i(j)$ all have values of 0.

The name proportional hazard stems from the fact that the hazard of any individual is a fixed proportion of the hazard of any other individual over time. The hazard ratio of two subjects with input vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ is

$$\frac{h(t, \mathbf{x}_1)}{h(t, \mathbf{x}_2)} = \frac{h_0(t) \exp(\boldsymbol{\beta}^T \mathbf{x}_1)}{h_0(t) \exp(\boldsymbol{\beta}^T \mathbf{x}_2)} = \exp[\boldsymbol{\beta}^T (\mathbf{x}_1 - \mathbf{x}_2)]. \tag{6.13}$$

This expression is independent of t meaning that the ratio of the hazards remains constant over time. By taking logarithms, we have

$$\log(h(t, \mathbf{x}_1)) - \log(h(t, \mathbf{x}_2)) = \boldsymbol{\beta}^T (\mathbf{x}_1 - \mathbf{x}_2). \tag{6.14}$$

Hence, if we plot $\log(h(t, \mathbf{x}_1))$ and $\log(h(t, \mathbf{x}_2))$ of two subjects, the curves will be parallel over time which implies that the subjects most at risk at any one time remain the subjects most at risk at all other times. From (6.13), it can also be seen that the inputs have a multiplicative impact on the hazard rate, since e.g. for all $\mathbf{x}_2(j) = 0$, we have

$$\frac{h(t, \mathbf{x}_1)}{h(t, \mathbf{x}_2)} = \exp(\boldsymbol{\beta}^T \mathbf{x}_1). \tag{6.15}$$

.

The $\boldsymbol{\beta}$ parameters of the proportional hazards model can be estimated without having to specify the baseline hazard function $h_0(t)$. Therefore, the proportional hazards model is often called a semi-parametric model. The estimation of the $\boldsymbol{\beta}$ coefficients is done by using the partial likelihood principle and ranking the event times of all non-censored subjects $t_{(1)} < t_{(2)} < ... < t_{(k)}$[3]. Suppose $\mathbf{x}_i$ denotes the vector of inputs of a subject with event time $t_{(i)}$. Given the fact that one subject

[3]The term partial likelihood refers to the fact that one only considers the event times and not the censoring times in the construction of the likelihood function.

has event time $t_{(i)}$, the probability that this subject has inputs $\mathbf{x}_i$ is then given by:

$$\frac{h(t_{(i)}, \mathbf{x}_i)\Delta t}{\sum_{l \in R(t_{(i)})} h(t_{(i)}, \mathbf{x}_l)\Delta t} = \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_i) h_0(t_{(i)})}{\sum_{l \in R(t_{(i)})} \exp(\boldsymbol{\beta}^T \mathbf{x}_l) h_0(t_{(i)})} = \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_i)}{\sum_{l \in R(t_{(i)})} \exp(\boldsymbol{\beta}^T \mathbf{x}_l)}, \tag{6.16}$$

whereby $R(t_{(i)})$ represents the subjects that are at risk at $t_{(i)}$.

The likelihood function then becomes:

$$\prod_{i=1}^{k} \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_i)}{\sum_{l \in R(t_{(i)})} \exp(\boldsymbol{\beta}^T \mathbf{x}_l)}, \tag{6.17}$$

where i runs over all event times. The $\boldsymbol{\beta}$ parameters are then chosen to optimize the logarithm of this partial likelihood function using the Newton-Raphson algorithm [3, 49, 133, 161]. Observe how the censored observations do enter the partial likelihood. These observations will be included in the risk sets $R(t_{(i)})$ until their censoring time. Equation (6.17) also indicates that the partial likelihood only depends on the order of the event times and not on their exact values.

It is important to note that the partial likelihood formula of (6.17) assumes that no two events occur at the same time. Many methods have been presented to deal with this problem of tied event times. An exact method considers that the ties arise from imprecise time measurements and assumes that there exists a true time ordering for the tied data. The likelihood function is then extended in order to take into account the various possible time orderings of the tied event times as illustrated in Example 6.2.

Example 6.2
Suppose that 3 events occur at time t_i for the subjects 1, 2 and 3 and n_i subjects are at risk that time. If the events take place in the order 1, 2, 3 then the partial likelihood for these 3 events would be

$$\frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_1)}{\sum_{j=1}^{n_i} \exp(\boldsymbol{\beta}^T \mathbf{x}_j)} \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_2)}{\sum_{j=2}^{n_i} \exp(\boldsymbol{\beta}^T \mathbf{x}_j)} \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_3)}{\sum_{j=3}^{n_i} \exp(\boldsymbol{\beta}^T \mathbf{x}_j)}. \tag{6.18}$$

If the events would occur in the order 2, 1, 3 the partial likelihood becomes

$$\frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_2)}{\sum_{j=1}^{n_i} \exp(\boldsymbol{\beta}^T \mathbf{x}_j)} \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_1)}{\sum_{j=1}^{n_i} \exp(\boldsymbol{\beta}^T \mathbf{x}_j) - \exp(\boldsymbol{\beta}^T \mathbf{x}_2)} \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_3)}{\sum_{j=3}^{n_i} \exp(\boldsymbol{\beta}^T \mathbf{x}_j)}. \tag{6.19}$$

All 3! possible orderings should then be considered and included in the partial likelihood formula of (6.17).

However, when the data is heavily tied, this procedure quickly becomes computationally intractable. Hence, approximations have been suggested in the literature in order to deal with this problem. The most popular are the Breslow [34] and Efron [70] approximations. The Breslow likelihood is

$$L_B = \prod_{i=1}^{k} \frac{\exp(\boldsymbol{\beta}^T \mathbf{s}_i)}{(\sum_{l \in R(t_{(i)})} \exp(\boldsymbol{\beta}^T \mathbf{x}_l))^{d_i}}, \tag{6.20}$$

whereby $\mathbf{s}_i$ is the sum of the values $\mathbf{x}_i$ of the subjects whose events occur at time $t_{(i)}$ and d_i is the number of subjects with events at that time. The Efron likelihood is

$$L_E = \prod_{i=1}^{k} \frac{\exp(\boldsymbol{\beta}^T \mathbf{s}_i)}{\prod_{j=1}^{d_i} [\sum_{l \in R(t_{(i)})} \exp(\boldsymbol{\beta}^T \mathbf{x}_l) - \frac{j-1}{d_i} \sum_{l \in D_i} \exp(\boldsymbol{\beta}^T \mathbf{x}_l)]}, \tag{6.21}$$

whereby D_i denotes the set of subjects with event times t_i. It is often stated that the Efron likelihood yields better parameter estimates than the Breslow likelihood [3].

The baseline hazard $h_0(t)$ can be estimated by using a non-parametric maximum likelihood approach [49, 133]. This estimate may then be used in combination with the estimated $\boldsymbol{\beta}$ parameters to compute hazard curves for individual subjects. Using (6.12) and (6.4), we can then derive the survival function for subject i:

$$S(t, \mathbf{x}_i) = [S_0(t)]^{\exp(\boldsymbol{\beta}^T \mathbf{x}_i)} \quad \text{with} \quad S_0(t) = \exp(-\int_0^t h_0(u) du) = \exp(-H_0(t)). \tag{6.22}$$

Note that it is also possible to assume a parametric baseline hazard $h_0(t)$. E.g., if we assume the baseline hazard to be exponential, the proportional hazards model becomes:

$$h(t, \mathbf{x}_i) = \lambda \exp[\boldsymbol{\beta}^T \mathbf{x}_i]. \tag{6.23}$$

The $\boldsymbol{\beta}$ and λ parameters can then be estimated in one joint maximum likelihood function [49, 133]. Furthermore, it can be shown that when the lifetime distribution is assumed Weibull (or exponential, since this is just a special case of Weibull), the proportional hazards model is at the same time an accelerated failure time model where the effect of the inputs is to accelerate (or decelerate) the time to failure [49, 133].

Two popular extensions of the proportional hazards model are time-varying inputs and stratification [3, 49, 133]. In many situations, it is convenient to have the inputs change over time. The only thing that changes in the model formulation is that the inputs are now indexed by time: $\mathbf{x}_i(t)$. When all inputs are time-varying, the Cox model becomes

$$h(t, \mathbf{x}_i(t)) = h_0(t) \exp[\boldsymbol{\beta}^T \mathbf{x}_i(t)]. \tag{6.24}$$

It is important to note that in this case the proportional hazards assumption no longer holds because the time-dependent inputs will change at different rates for different subjects. Note that although the inputs are time-varying, the $\boldsymbol{\beta}$

parameters remain constant over the time period considered. If the $\boldsymbol{\beta}$ parameters were also allowed to vary over time, one obtains the most general version of the proportional hazards model

$$h(t, \mathbf{x}_i(t)) = h_0(t) \exp[\boldsymbol{\beta}^T(t)\mathbf{x}_i(t)]. \tag{6.25}$$

Another way to introduce non-proportionality is stratification whereby different baseline hazards $h_{0s}(t)$ are assumed for different subject populations (strata) $s = 1, ..., S$ that have different values for a stratification variable. This can be useful when the data contains different clusters each having their own specific baseline hazard function. Again, the $\boldsymbol{\beta}$ parameters may be estimated by using a straightforward extension of the partial likelihood principle.

6.2.4 Discrete Proportional Hazards Models

When the time variable T is discrete, both the Breslow and Efron approximations of (6.20) and (6.21) may yield poor estimates because of the substantial number of ties per time point. In this subsection, we will discuss briefly two extensions of the proportional hazards model for discrete survival time data.

Let T be a random variable with values $t_1 < t_2 < ...$ and corresponding probabilities:

$$f(t_i) = f_i = P(T = t_i). \tag{6.26}$$

The survivor function is defined as follows:

$$S(t_i) = S_i = P(T \geq t_i) = \sum_{k=i}^{\infty} f_k, \tag{6.27}$$

whereas the hazard function now becomes

$$h(t_i) = h_i = P(T = t_i | T \geq t_i) = \frac{f_i}{S_i}. \tag{6.28}$$

Note that the hazard is now a conditional probability rather than a rate. In order to survive time period t_i, a subject must first survive t_1, then t_2 given that he survived t_1 and so on, which gives

$$S_i = (1 - h_1)(1 - h_2)...(1 - h_{i-1}), \tag{6.29}$$

and thus also,

$$\frac{S_{i+1}}{S_i} = 1 - h_i. \tag{6.30}$$

A discrete analogue of the proportional hazards model of (6.12) can then be obtained by starting from (6.22)

$$S(t, \mathbf{x}_i) = [S_0(t)]^{\exp(\boldsymbol{\beta}^T \mathbf{x}_i)}. \tag{6.31}$$

Since, according to (6.29)

$$S_0(t) = \prod_{t_j < t} (1 - h_0(t_j)), \tag{6.32}$$

and recalling (6.30), we obtain the following relationship for the hazard

$$h(t, \mathbf{x}_i) = 1 - [1 - h_0(t)]^{\exp(\boldsymbol{\beta}^T \mathbf{x}_i)}, \tag{6.33}$$

or

$$1 - h(t, \mathbf{x}_i) = [1 - h_0(t)]^{\exp(\boldsymbol{\beta}^T \mathbf{x}_i)}. \tag{6.34}$$

The $\boldsymbol{\beta}$ parameters may then be estimated using a maximum likelihood procedure [182]. Hereto, one applies a cloglog-transformation in order to improve the Newton-Raphson convergence

$$\log(-\log(1 - h(t, \mathbf{x}_i))) = \alpha_t + \boldsymbol{\beta}^T \mathbf{x}_i, \tag{6.35}$$

with $\alpha_t = \log(-\log(1 - h_0(t)))$. It is interesting to note that this model can also be obtained by grouping time in the continuous-time proportional hazards model of (6.12) [133].

Another discrete survival analysis model has been proposed by Cox and specifies a linear log odds relationship between the discrete hazard and the inputs[49]

$$\frac{h(t, \mathbf{x}_i)}{1 - h(t, \mathbf{x}_i)} = \frac{h_0(t)}{1 - h_0(t)} \exp(\boldsymbol{\beta}^T \mathbf{x}_i), \tag{6.36}$$

or equivalently

$$\log(\frac{h(t, \mathbf{x}_i)}{1 - h(t, \mathbf{x}_i)}) = \alpha_t + \boldsymbol{\beta}^T \mathbf{x}_i, \tag{6.37}$$

with $\alpha_t = \log(\frac{h_0(t)}{1-h_0(t)})$ and $h(t, \mathbf{x}_i)$ and $h_0(t)$ the discrete (baseline) hazard. The α_t typically differ for each time interval. This model is often referred to as the *proportional odds model for grouped survival times.* Note the similarity between (6.37) and an ordinary logistic regression model. It is precisely this analogy that allows to use the ordinary logistic regression estimation procedure on a slightly transformed data set to estimate the parameters $\boldsymbol{\beta}$ and α_t in a maximum likelihood way [49]. This model is most appropriate when events can only occur at regular, discrete points in time whereas the cloglog-model of (6.35) is more suited when ties arise from grouping continuous-time data into intervals [3].

6.3 Neural Networks for Survival Analysis

A first drawback of the proportional hazards models discussed in the previous section, is that the functional form of the inputs remains linear or some mild extension thereof. If more complex terms are to be included (e.g. interaction terms between inputs, quadratic terms, ...), they must be specified somewhat arbitrarily by the user [58]. Furthermore, this linear form invokes extreme hazard rates for subjects with outlying values for their inputs [159]. And finally, in the standard proportional hazards model, the baseline hazard function is assumed to be uniform across the entire population resulting in proportional hazards. Although time-varying inputs and stratification allow for non-proportionality, these extensions might not provide the best way to model the baseline variation [58, 159]. In this section, we will discuss how neural networks might offer an answer to these problems by reviewing a number of studies using neural networks for survival analysis.

6.3.1 Direct Classification

The simplest method considers survival for a fixed time period, and consequently gives a binary classification problem [30, 40, 233]. Censored observations are removed and biases are introduced. The neural network output then provides an estimate of the probability that a subject will survive the time period. Above the 50% threshold, the subject is assumed to survive the period. It is clear that this approach is rather basic and does not allow to produce individual survival or hazard curves. Furthermore, it does not deal with the problem of censoring and time-varying inputs.

6.3.2 Ohno-Machado

Ohno-Machado [174, 175] uses multiple neural networks to solve the survival analysis problem. Each neural network has a single output predicting survival at a certain time point. The networks are then trained using their own data subsets consisting of cases that made it to the corresponding time period. Censored observations are included until their time of censoring. Hence, the number of training instances gradually decreases for the later time intervals making the predictions less reliable. The author argues that when using these neural networks in isolation, non-monotonic survival curves may result. As a result, the probability of a person surviving two periods could be greater than the probability to survive one period because the interdependencies of the survival probabilities over time are not properly taken into account when isolated neural networks are used. The author describes a way to decrease the frequency of non-monotonic curves by combining the neural networks. Survival predictions of one neural network are then used as an additional input to another neural network as illustrated in Figure 6.3. However, it is still possible to obtain non-monotonic survival curves although the departure from monotonicity is smaller than if the neural networks were not connected to each other. Furthermore, the issue of how to combine the neural

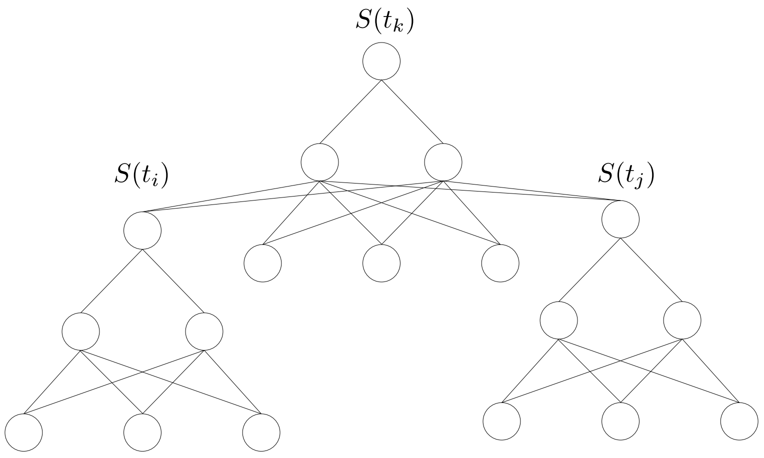


Figure 6.3: An example of a modular neural network for survival analysis whereby the output of the networks predicting $S(t_i)$ and $S(t_j)$ are used as additional inputs for the network predicting $S(t_k)$.

networks remains an open question. Although not presented in the original paper, the approach allows to easily include time-dependent inputs into the different data subsets. However, the necessity to use multiple neural networks and the question how to combine them represent an important scalability problem which makes the method less suitable for handling large data sets.

6.3.3 Ravdin and Clark

Ravdin and Clark [57, 58, 189] use a multi-layer feed-forward neural network with a single output unit representing the survival status. A time indicator and a survival status indicator are added to each record. The time indicator then records the successive time periods $[1, T_{max}]$ for which a prediction is to be made, with T_{max} the maximum time of follow-up. An uncensored input is then replicated T_{max} times whereas a censored input is replicated t times with t being the time of censoring. The survival status is the target of the network and is set to zero as long as the subject is alive and to 1 otherwise. This is illustrated in Example 6.3.

Example 6.3
If a subject with attributes a and b has died at time 3 with T_{max}=5 then the following records are created (a,b,1,0), (a,b,2,0), (a,b,3,1), (a,b,4,1), (a,b,5,1). If a subject with attributes c and d is censored at time 4 with $T_{max} = 5$ the following records are created (c,d,1,0),(c,d,2,0), (c,d,3,0),(c,d,4,0).

Although time dependent inputs were not discussed in the original study, they can be easily included into the corresponding data records. The authors state that the output of the neural network (referred to as the prognostic index) is roughly proportional to the Kaplan-Meier estimate of the survival probability. However, they provide no guarantees that the generated survival curves will be monotonically decreasing. Furthermore, the replication of records introduces two problems. First, it will result in large biases because the number of deaths in the late time intervals will be overrepresented. The authors suggest to handle this by selective sampling such that the proportion of deaths matches the Kaplan-Meier estimate. Second, while this method does not require the use of multiple networks, it will result in very large data sets, which causes severe scalability problems.

6.3.4 Biganzoli et al.

A variation on the approach of Ravdin and Clark was suggested by Biganzoli et al. [25]. They also train a neural network with one output and an additional time indicator input. However, unlike Ravdin and Clark, uncensored subjects are only replicated for the time intervals in which they were actually observed. Hence, subjects that have died are not included after the time interval of death. Again, time dependent inputs might be easily included since each subject has multiple input vectors which may change across the intervals of observation. The neural network predicts discrete hazard rates which may be easily converted to monotone survival probabilities using (6.29). However, the approach is not scalable because of the enormous data replication requirements.

6.3.5 Lapuerta et al.

Lapuerta et al. [144] suggest a multi-network strategy to impute the survival times for the censored cases. For each time period considered, a separate neural network is constructed. These networks are trained using only the observations for which the survival status for the corresponding time period is known. Subsequently, the trained networks are used to predict the outcome for the censored cases. The non-censored and imputed censored observations are then provided for training the principal neural network (referred to as the Predictor network in the original paper) which predicts the probability of survival for each time period considered. Although the proposed method compares favorably to the Cox proportional hazards model, no guarantees are provided that the derived survival probabilities are monotonically decreasing and time varying inputs are also not allowed. Furthermore, it is clear that this approach is not suitable for large-scale applications since one needs to train as many neural networks as there are time periods considered.

6.3.6 Faraggi

Faraggi [78, 79, 160] proposes a neural network extension of the Cox proportional hazards model by replacing the linear function $\boldsymbol{\beta}^T \mathbf{x}_i$ in (6.12) by the output $g(\mathbf{x}_i, \boldsymbol{\theta})$ of a neural network with a single, logistic hidden layer and a linear output layer

$$h(t, \mathbf{x}_i) = h_0(t) \exp[g(\mathbf{x}_i, \boldsymbol{\theta})]. \tag{6.38}$$

Analogous to the Cox model, no bias input is considered for the output layer since this is implicitly incorporated into the baseline hazard $h_0(t)$. The $\boldsymbol{\theta}$ parameters are then also estimated using the partial likelihood principle and Newton-Raphson optimization. The approach was applied in a breast cancer study in [160]. This method allows to preserve all the advantages of the classical proportional hazards model. However, the standard approach still assumes that the hazards are proportional. Although time-varying covariates and/or stratification might allow for non-proportionality, these extensions may not be the best way to model the baseline variation.

6.3.7 Street

Street [223] uses a multilayer perceptron with T_{max} output units to tackle the survival analysis problem, whereby T_{max} represents the maximum time horizon of the study. A hyperbolic tangent activation function is used in the output layer such that all output neurons take on values between -1 and $+1$. The first output neuron having a value < 0 is considered to be the output neuron that predicts the event time. If all output neurons have values > 1, then the patient is considered to survive the entire time period of the study. The output units thus represent the survival probability for the corresponding time period.

For the non-censored cases, the output values are set to $+1$ as long as the patient is alive and to -1 thereafter. For the censored cases, the output units are also set to $+1$ until their censoring time. After this period, Street uses the Kaplan-Meier estimates of (6.6)

$$S(t) = S(t-1) \times (1 - h(t)), \tag{6.39}$$

with $h(t) = \frac{d_t}{n_t}$, whereby d_t represents the number of deaths in period t, and n_t represents the subjects at risk in that period. The latter number is calculated by subtracting from the number of subjects at risk at the beginning of period $t-1$, the total number of deaths and the total number of censored observations in that same period. The Kaplan-Meier hazards are then used to compute the survival probability of the censored observations after the censoring time. Note that these probabilities are then scaled to the range of the hyperbolic tangent function in the following way: $\textit{activation} = 2 \times \textit{probability} - 1$.

Example 6.4
Consider a study with 100 subjects. In the first time interval, 6 subjects die and 4 are censored. In the second time interval, 10 subjects die. Hence, we have $S_1 = 1 - 6/100 = 0.94$ and $S_2 = S_1(1 - 10/90) = 0.8355$ since there are 90 patients at risk during the second time interval.

In summary, the outputs of the training set observations are encoded as follows:

$$S(t) = \begin{cases} 1 \quad 1 \leq t \leq L \\ -1 \quad D = 1 \ and \ L < t \leq T_{max} \\ S(t-1) \times (1 - h(t)) \quad D = 0 \ and \ L < t \leq T_{max}, \end{cases} \tag{6.40}$$

whereby T_{max} represents the maximum number of time periods involved in the study, L the subject lifetime or censoring time, and D indicates if the subject is censored ($D = 0$) or not ($D = 1$). The individual survival curve of an observation can then be derived based on the activation values of the output units. Since the neural network cannot be forced to generate monotonically decreasing output units, a non-monotone survival curve is still possible, which complicates its interpretation [159]. Furthermore, no extension is provided to deal with time-varying inputs.

6.3.8 Mani

A variation on the method of Street was developed by Mani [159]. Again, for every observation in the training set, T_{max} output units are computed. Nevertheless, these output units now represent the hazard rate instead of the survival probabilities that were used in the approach of Street. The outputs are then computed as follows:

$$h(t) = \begin{cases} 0 \quad 1 \leq t \leq L \\ 1 \quad D = 1 \ and \ L < t \leq T_{max} \\ \frac{d_t}{n_t} \quad D = 0 \ and \ L < t \leq T_{max} \end{cases} \tag{6.41}$$

Again, T_{max} represents the maximum number of periods involved in the study, L the subject lifetime or censoring time, and D indicates if the subject is censored ($D = 0$) or not ($D = 1$). For uncensored observations, the hazard is set to zero until the time of death and 1 thereafter. For censored observations, the hazard is set to zero until censoring time and to the Kaplan-Meier estimate thereafter. The survival probabilities may then be estimated by using (6.6). The generated survival curves will thus be monotonically decreasing which simplifies the interpretation and increases robustness [159]. However, the topic of time-varying inputs has been left unaddressed.

6.3.9 Brown et al.

Analogous to Mani, Brown suggests a single neural network with multiple outputs to predict hazard rates [39]. For the non-censored observations, the network output is set to 0 as long as the subject is alive and to 1 when the subject undergoes the event. For the time intervals following the event, the hazard is unconstrained. The output values for the censored observations are set to 0 until the time of censoring and are unconstrained for all subsequent time intervals. The authors then suggest to train the neural network to minimize the sum of squared error criterion and to perform no weight updates when the hazard is unconstrained by setting the corresponding errors to 0. The approach presented is scalable and results in monotonic survival curves. Again, no extension is presented to deal with time-varying inputs.

6.3.10 Discussion

Table 6.1 presents an overview of the characteristics of the neural network based methods for survival analysis discussed in the previous subsections. From the literature review above, it becomes clear that for large scale data sets, the approaches of Faraggi, Mani and Brown seem the most interesting. All three allow to generate monotonically decreasing survival curves and only one neural network needs to be trained. Although the approach of Faraggi also allows for time-varying inputs, it is less flexible in modeling the baseline variation. On the other hand, while the approaches of Mani and Brown allow for flexible baseline modeling, they do not solve the problem of time-varying inputs. Note that this literature review on the use of neural networks for survival analysis is by no means exhaustive. Other interesting references are, e.g., [19, 150, 194, 195].

6.4 Survival Analysis for Credit Scoring

Traditionally, the primary goal of credit scoring was to distinguish good customers from bad customers without taking into account when customers tend to default. The latter issue is however becoming more and more a key research question since the whole process of credit granting and repayment is being more and more conceived as dynamic instead of static. The advantages of having models that estimate when customers default are [21, 235]

- the ability to compute the profitability over a customer's lifetime and perform profit scoring;
- these models may provide the bank with an estimate of the default levels over time which is useful for debt provisioning;

	Multiple NN	**Single Output**	**Monotone survival curve**	**Censoring**	**Time-varying covariates**	**Scalable**
Direct classification [30, 40, 233]	N	Y	N	N	N	Y
Ohno-Machado [174, 175]	Y	Y	N	Y	Y	N
Ravdin and Clark [57, 58, 189]	N	Y	N	Y	Y	N
Biganzoli et al. [25]	N	Y	Y	Y	Y	N
Lapuerta et al. [144]	Y	N	N	Y	N	N
Faraggi [78, 79, 160]	N	Y	Y	Y	Y	Y
Street [223]	N	N	N	Y	N	Y
Mani [65, 159]	N	N	Y	Y	N	Y
Brown [39]	N	N	Y	Y	N	Y

Table 6.1: Characteristics of neural network survival analysis methods.

- the estimates may help to decide upon the term of the loan;
- changes in economic conditions can be easier incorporated.

Until now, not many authors have addressed the issue of predicting customer default times. In what follows, we will present a short literature overview on this topic.

Narain was one the first authors to use survival analysis methods for credit scoring [169]. He analysed a data set of 1242 applicants accepted for a 24 month loan between mid 1986 and mid 1988. Each loan was described by 7 characteristics of the applicant. A loan was considered bad if three consecutive payments have been missed. The author then tries to predict the time until a loan defaults. All good loans are considered to be right censored. Of the 1242 applicants, 533 were bad and 709 right censored. The data was analysed using the Kaplan-Meier method and by fitting exponential regression models. It was shown that the results obtained are encouraging and reasonable.

Banasik et al. report on the use of the proportional hazards model for predicting when borrowers default [21]. They use personal loan data from a major U.K. financial institution which consists of application information of 50000 loans accepted between June 1994 and March 1997 together with their monthly performance description for the period up to July 1997. The data set was randomly split into a training set (70% of the observations) and a test set (30% of the observations). Motivated by the competing risks approach, the authors suggest to conduct the survival analyses for the bad and early repaid loans separately. For the former, all bad loans are considered to be failed and the others censored, whereas in the latter case all loans that are paid off early are treated as failed and all others censored. It is then investigated how likely the loans are to become bad or paid off early in their first 12 months and between 12 and 24 months. Hereto, the authors use the non-parametric proportional hazards model (no baseline hazard assumption), two parametric proportional hazards models using exponential and Weibull baseline hazards, and an ordinary logistic regression approach. It is shown that, for the first year, the proportional hazards models are competitive with the logistic regression model for predicting default and may be superior for predicting early pay-off. However, for the second year, the results for the proportional hazards models are less encouraging when compared to the logistic regression models.

Stepanova and Thomas continue the research by Banasik et al. and try to further augment the performance of the estimated proportional hazards models [222]. They propose a new method to segment continuous or group discrete variables based on inspecting the survival analysis parameter estimates for the different values of the variable. It is argued that this procedure should be done separately for each failure type (early repayment or default). It is also suggested to segment the data according to the term of the loans and estimate separate models for each segment. The proportional hazards and logistic regression models are compared with respect to the classification accuracy and the ROC curve. Furthermore, the model fit of the proportional hazards models is also assessed by inspecting the Cox-snell, Martingale, and Schoenfeld residuals. Finally, the authors also propose to include time-dependent inputs in order to overcome the proportionality assumption.

In [221], Stepanova and Thomas build behavioral scoring models using proportional hazards analysis. They used data provided by a U.K. financial institution consisting of 11500 customers with 16 application characteristics and 4 performance variables measured during 36 months. The data was split into three samples (two training samples and one hold-out sample) of approximately equal size. The first sample was used to build an application score using stepwise proportional hazards regression. PHAB (proportional hazards analysis behavior scores) models where then built using the second training sample for each month of the life of a loan. The predictors were the estimated application score and the performance variables. Several combinations of predictors were tested out. Comparisons are made with logistic regression by inspecting the ROC curves. It is also illustrated how the PHAB scores can be used to compute the expected profit during each month of the loan (profit scoring). The authors conclude by saying that the PHAB scores are useful as indicators of both risk and profit.

6.5 Empirical Evaluation

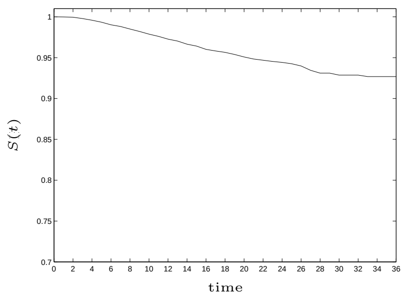
6.5.1 Experimental Setup and Data Set Characteristics

The statistical and neural network survival analysis techniques were applied to personal loan data from a major U.K. financial institution [12]. All customers are U.K. borrowers who had applied to the bank for a loan. The data set consisted of the application information of 50000 personal loans, together with the repayment status for each month of the observation period of 36 months. Application characteristics available in the data set are summarized in Table 6.2. The status variable indicated which loans were bad, paid off to term, paid off early, or still open. We note that the same data was also used in [222]. However, we took a subsample of 15000 observations and only considered loans having a duration of less than 36 months. Missing values were imputed using the mean for the continuous attributes and the most frequent category for the categoric attributes. The data was randomly split into a training set (10000 observations) and a test set (5000 observations). Table 6.3 describes the various purposes of the loans.

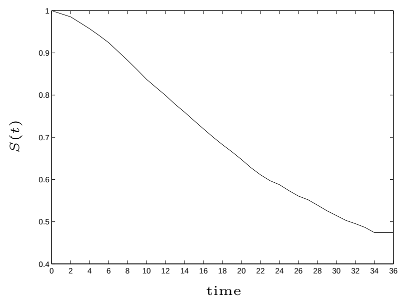
Figure 6.4 depicts the Kaplan-Meier curves for both loan default and early repayment. Note that in the default case, the Kaplan-Meier curve is very flat at the beginning since our definition of a default is three months of payments missed and thus no defaults occur during the first three months.

Number	Characteristic
1	Customer Age
2	Amount of Loan
3	Years at Current Address
4	Years with Current Employer
5	Customer Gender
6	Number of Dep. Children
7	Frequency paid
8	Home Phone Number Given
9	Insurance Premium
10	Loan type (single or joint)
11	Marital Status
12	Term of Loan
13	Home Ownership
14	Purpose of Loan

Table 6.2: Data set characteristics.



(a) KM curve for default



(b) KM curve for early repayment

Figure 6.4: Kaplan Meier curves for default and early repayment.

In the following subsections, we will investigate the use of statistical and neural network survival analysis techniques for predicting both early repayment and loan default. For the former, we considered all loans that are paid off early as failures and all other loans as censored whereas in the latter case all defaulted loans are failures and the remaining ones censored [21, 222]. For the statistical approaches, we will experiment with the standard Cox model of 6.12 using both the Breslow (6.20) and the Efron (6.21) approximations, as well as with its discrete analogue variant of 6.36. The statistical survival analysis approaches will be implemented using proc phreg in SAS^{TM} [3].

Number	**Purpose**
1	Account standard
2	Caravan
3	New Car
4	Car Repair
5	Electrical
6	General Living
7	Home Improvement
8	Honeymoon
9	Motor Caravan
10	Mixed Purchases
11	Others
12	Redecoration
13	Remortgages
14	Weddings
15	Boat
16	Motor Cycle
17	Car Over 3Yr Old
18	Car Under 3Yr Old
19	Furniture
20	Graduate Loan
21	Holiday
22	Kitchen Units
23	Musical Instrument
24	Other Specific
25	Other Vehicles
26	Refinance
27	Van

Table 6.3: The purpose attribute.

For the neural network analyses, we will adopt a variant of the approach suggested by Mani (see subsection 6.3.8). In order to generate monotonically decreasing survival curves, we will train the neural network to predict hazard rates and transform these to survival probabilities using equation 6.6. Figure 6.5 depicts how the outputs of the neural network are encoded. For the good customers, the output is set to zero until the last point of observation, to the Kaplan-Meier hazard until the term of the loan, and to 0 until the last time period considered. For the bad customers, we set the output to zero until the time of loan default and to 1 for all subsequent time periods. Concerning the architecture of the neural network, we use one hidden layer influenced by theoretical works with show that NNs with one hidden layer are universal approximators capable of approximating any continuous function to any desired degree of accuracy on a compact interval[26].

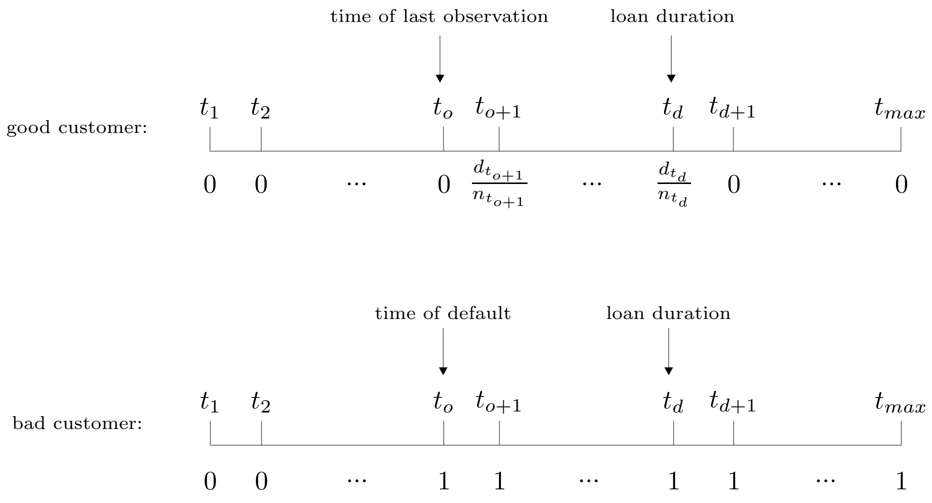

time of last observation
loan duration
good customer:
t_1 t_2 t_o t_{o+1} t_d t_{d+1} t_{max}
0 0 ··· 0 $\frac{d_{t_{o+1}}}{n_{t_{o+1}}}$ ··· $\frac{d_{t_d}}{n_{t_d}}$ 0 ··· 0
time of default
loan duration
bad customer:
t_1 t_2 t_o t_{o+1} t_d t_{d+1} t_{max}
0 0 ··· 1 1 ··· 1 1 ··· 1

Figure 6.5: Encoding of the neural network outputs for survival analysis.

The hidden neurons have hyperbolic tangent transfer functions and the output neurons logistic transfer functions. The number of hidden neurons is determined by experimental evaluation. Although some authors advocate the use of a sum of squares error function (e.g. [39]), the cross-entropy error function is the most popular in the neural network survival analysis literature (e.g. [159, 175, 223]). Hence, we train the networks to minimize a regularized cross-entropy error function as follows [15, 26]

$$G = -\sum_{i=1}^{N} y_i \log(z_i) + (1 - y_i)\log(1 - z_i) + \sum_k \alpha_k E_{W(k)}, \tag{6.42}$$

where y_i is the target output, z_i is the network output and $E_{W(k)}$ a regularization term representing half of the squared weights of weight class k. Note that we introduce $n + 3$ weight regularization terms $E_{W(k)}$, one for each input, one associated with the input bias neuron, one with the hidden layer connections and one with the hidden layer bias neuron. It is well-known that introducing regularization terms into the neural network objective function is an efficient way to avoid overfitting. The α_k parameters are then optimized on-line using the automatic relevance determination (ARD) extension of the Bayesian evidence framework of MacKay [15, 154, 155]. This framework assumes that all weights of weight class k are distributed according to a Gaussian prior with mean 0 and $\sigma_k^2 = \frac{1}{\alpha_k}$. The inferred α_k values may then be used to order the inputs with the most relevant inputs having the lowest α_k values (for more details see [154, 155]). All neural network analyses are conducted using the Netlab toolbox^{TM} [168] (`http://www.ncrg.aston.ac.uk/netlab/`).

Measuring the performance of a survival analysis model is not a trivial exercise.

Following Banasik et al. [21] and Stepanova and Thomas [222], we will compare the performance of the survival analysis models by looking at the following criteria:

1. Estimating which loans will be paid off early or default within the first 12 months.
2. Estimating which loans, which are still repaying after 12 months, will pay off early or default within the next 12 months.

Remember that the proportional hazards assumption assumes that the customers most at risk at any one time remain the customers most at risk at all other times. Note that this does not automatically imply that the same customers will be labelled as failed (paid off early or defaulted) under each of the above criteria since for the second criterion some of the customers that failed in the first 12 months will not be considered [21].

6.5.2 Results for Predicting Early Repayment

Before starting the analysis, we first grouped the categorical purpose attribute into three categories by considering the probability of early repayment for each loan purpose (see Table 6.4). The loan purposes having lowest probability of early repayment were categorized as low risk, those having medium probability of early repayment as medium risk, and the rest as high risk. Each category is then encoded using binary dummy variables. We compare the performance of the statistical and neural network survival analysis models with the performance of a logistic regression classifier. Taking into account the performance criteria discussed in the previous subsection, we estimate two logistic regression classifiers: one where the bads are all the loans that defaulted in the first 12 months and one where only loans that survived the first 12 months are considered and the bads are the ones that defaulted before 24 months. Table 6.5 presents the confusion matrix numbers of the logistic regression model, the Cox proportional hazards model and the neural network. For the logistic regression classifier, we chose the cut-off to map the output probabilities to class labels such that the predicted number of early repayments equals the actual number of early repayments. For the Cox and neural network survival analysis models, we labelled the observations having the lowest S(12) as early repaid again taking care that the predicted number of early repayments equals the actual number of early repayments. For the Cox model, we experimented with the standard Cox model of 6.12 using both the Breslow (6.20) and the Efron (6.21) approximations, as well as with its discrete analogue variant of 6.36. All models were significant according to the likelihood ratio and Wald test and the estimated parameters and p-values varied only very slightly. Hence, we report the results for the discrete model. The neural network trained had one hidden layer with 10 hidden neurons.

Low Risk	**Medium Risk**	**High Risk**
Graduate Loan	Home Improvement	Car Repair
Holiday	Other Specific	Car Over 3Yr Old
Van	Boat	General Living
Electrical	Others	Remortgages
New Car	Caravan	Mixed Purchases
Redecoration	Car Under 3Yr Old	Musical Instrument
Honeymoon	Weddings	Motor Cycle
Kitchen Units	Refinance	Motor Caravan
Other Vehicles	Furniture	Account standard

Table 6.4: Grouping the purpose attribute for predicting early repayment.

	Actual	Logit	Cox	NN
G-predicted G	4020	3247	3263	3279
G-predicted B	0	773	757	741
B-predicted G	0	773	757	741
B-predicted B	980	207	223	239
Attr. most imp.		term	term	insurance premium
Attr. 2nd most imp.		amount	low risk purp	medium risk purp
Attr. 3rd most imp.		low risk purp	years at address	high risk purp

Table 6.5: Predicting early repayment in first 12 months.

It can be observed from Table 6.5 that the logistic regression classifier achieved a PCC of 69.08%, the Cox model a PCC of 69.72% and the neural network a PCC of 70.36% on the test set. The logistic regression and Cox model consider the term input as the most important whereas the neural network ranks the insurance premium as most relevant.

Table 6.6 reports the results for predicting early repayment between 12 and 24 months. For the survival analysis models, we hereto calculated $S(24)/S(12)$ and labelled the observations with the lowest value for this ratio as early repaid between 12-24 months again respecting the sample proportions. For the second

	Actual	Logit	Cox	NN
G-predicted G	2074	1661	1612	1639
G-predicted B	0	413	462	435
B-predicted G	0	413	462	435
B-predicted B	557	144	95	122

Table 6.6: Predicting early repayment 12-24 months.

performance measure, the logistic regression classifier achieved a PCC of 68.60%, the Cox model a PCC of 64.88% and the NN a PCC of 66.93%. This indicates that the performance benefit of the neural network when compared to the Cox model is more pronounced for the second criterion than for the first criterion. Although, for the second criterion, it is inferior when compared to the logistic regression classifier, it has to be mentioned that the comparison is not completely fair since two separate logistic regression classifiers were trained each specifically tailored to the desired performance objective. Furthermore, also remember that the logistic regression classifier only provides a classification decision whereas the Cox and neural network models also provide information upon the timing of the event.

To investigate the effect of each continuous variable, surface plots were generated from the neural network by fixing the remaining continuous variables to their median values and the categorical variables to their modal category. In general, the survival probability for early repayment increases with decreasing customer age (Figure 6.6 (a)), it showed an increasing trend with increasing amount (Figure 6.6 (b)), increasing years at address (Figure 6.6 (c)), increasing years with employer (Figure 6.6 (d)), increasing insurance premium (Figure 6.6 (e)) and increasing term (Figure 6.6 (f)). It has to be noted that these plots only provide a preliminary insight into the relationship between the survival probabilities and the inputs since non-additive interaction effects may exist between the inputs. However, they may allow the credit scoring expert to get an initial insight into the impact of the application characteristics on the survival probabilities.

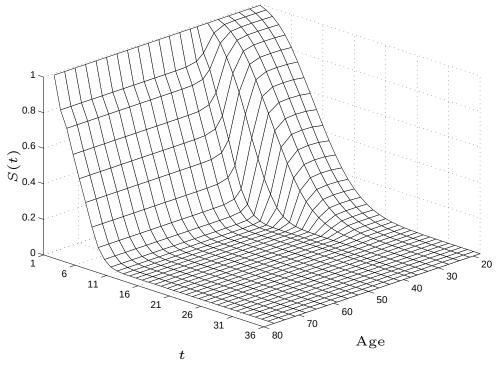
6.5.3 Results for Predicting Default

Since the most risky purposes for predicting early repayment, are not necessarily the same as for predicting default, we again start by dividing the purpose attribute into 3 categories (see Table 6.7). Table 6.8 presents the confusion matrix numbers

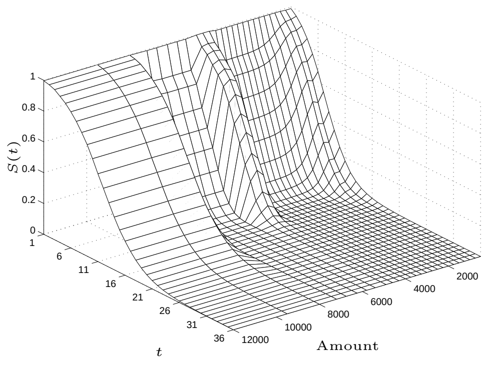
Low Risk	**Medium Risk**	**High Risk**
Account standard	Electrical	Mixed Purchases
Honeymoon	Caravan	Furniture
Motor Caravan	Weddings	Car Under 3Yr Old
Boat	New Car	General Living
Graduate Loan	Home Improvement	Other Vehicles
Kitchen Units	Other Specific	Car Repair
Van	Others	Refinance
Motor Cycle	Musical Instrument	Redecoration
Holiday	Car over 3Yr Old	Remortgages

Table 6.7: Grouping the purpose attribute for predicting default.

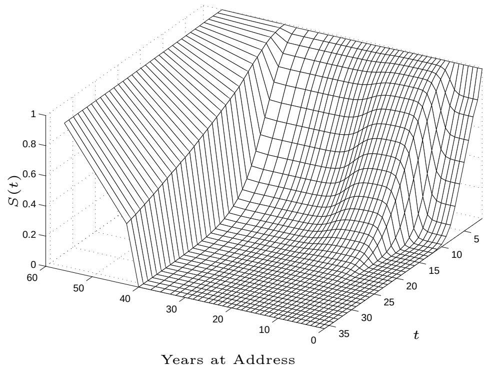
for predicting default in the first 12 months. Again, the Breslow, Efron and discrete versions of the Cox model gave significant p-values for the Likelihood ratio and



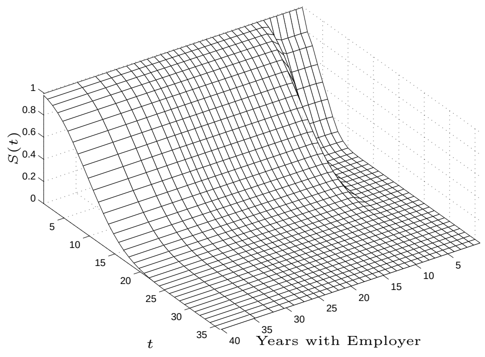
(a) Age



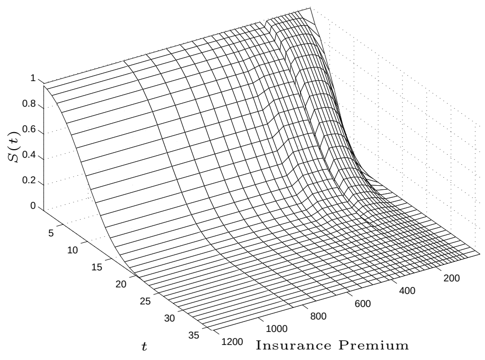
(b) Amount



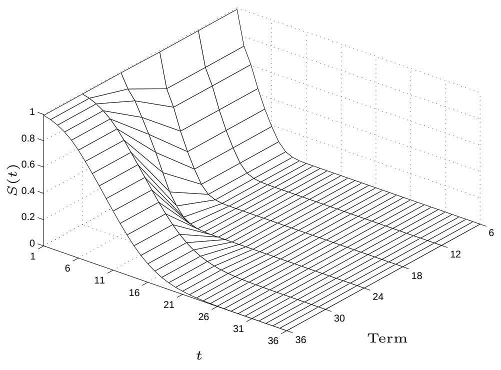
(c) Years at Address



(d) Years with Employer



(e) Insurance Premium



(f) Term

Figure 6.6: Evolution of neural network survival distributions with respect to inputs for predicting early repayment.

Wald test and yielded similar coefficients and p-values. The neural network had 8 hidden neurons. The logistic regression classifier and the Cox model gave the

	Actual	Logit	Cox	NN
G-predicted G	4870	4750	4750	4752
G-predicted B	0	120	120	118
B-predicted G	0	120	120	118
B-predicted B	130	10	10	12
Attr. most imp.		medium risk purp	medium risk purp	term
Attr. 2nd most imp.		low risk purp	low risk purp	medium risk purp
Attr. 3rd most imp.		years employed	years employed	years employed

Table 6.8: Predicting default in first 12 months.

same results (PCC=95.20%). The neural network only performed slightly better (PCC=95.28%). Note that the logistic regression classifier and the Cox model ranked the same 3 attributes as most important. Table 6.9 presents the results for predicting default in 12-24 months. The Cox and NN model yielded the same

	Actual	Logit	Cox	NN
G-predicted G	2567	2506	2509	2509
G-predicted B	0	61	58	58
B-predicted G	0	61	58	58
B-predicted B	64	3	6	6

Table 6.9: Predicting default 12-24 months.

performance (PCC=95.59%) which was marginally better than the performance of the logistic regression classifier (PCC=95.36%). When contrasting the results depicted in Tables 6.8 and 6.9 with those of Tables 6.5 and 6.6, it becomes clear that, for predicting default, the superiority of the neural network model is less pronounced than for predicting early repayment. One of the reasons behind this phenomenon might be that in the default case, the data is more skewed since only 130 customers of the 5000 actually defaulted in the first twelve months (2.6%) and only 64 of the 2631 defaulted between 12 and 24 months (2.4%). For that reason, we also conducted the experiments on a second default data set whereby we oversampled the number of defaults. Table 6.10 presents the results for predicting default in the first 12 months on the oversampled data set. The neural network had 16 hidden neurons. The logistic regression classifier yielded the best performance (PCC=79.20%) followed by the Cox model (PCC=79.0%) and the neural network (PCC=78.76%)). Observe how all three models agree on the importance of the number of years employed in predicting loan default. Table 6.11 gives the results for loan default between 12 and 24 months. Here, the neural network was superior and yielded a PCC of 78.58% whereas the logistic regression classifier gave a PCC of 78.24% and the Cox model a PCC of 77.50%.

Analogous to the previous subsection, we also generate 3D surface plots from

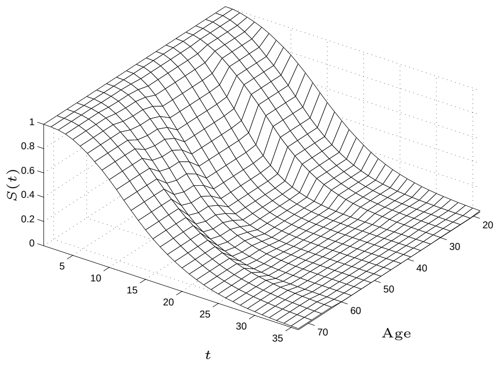
	Actual	Logit	Cox	NN
G-predicted G	4208	3688	3683	3677
G-predicted B	0	520	525	531
B-predicted G	0	520	525	531
B-predicted B	792	272	267	261
Attr. most imp.		years employed	years employed	years employed
Attr. 2nd most imp.		medium risk purp	medium risk purp	insurance premium
Attr. 3rd most imp.		insurance premium	insurance premium	frequency paid

Table 6.10: Predicting default in first 12 months on oversampled data set.

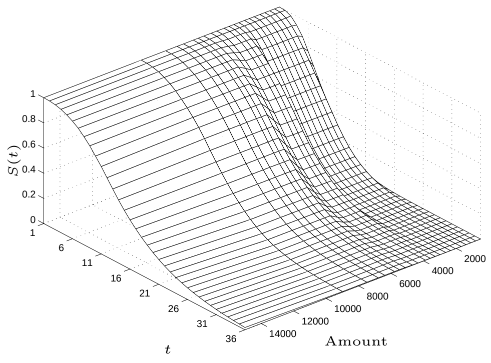
	Actual	Logit	Cox	NN
G-predicted G	2015	1753	1744	1757
G-predicted B	0	262	271	258
B-predicted G	0	262	271	258
B-predicted B	394	132	123	136

Table 6.11: Predicting default 12-24 months on oversampled data set.

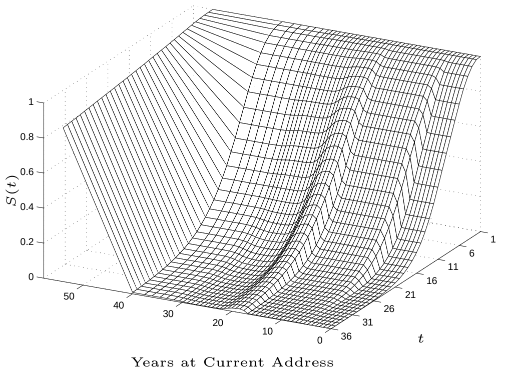
the neural network outputs in order to present a general view of the sensitivity of the survival probabilities with respect to the continuous inputs (see Figure 6.7).



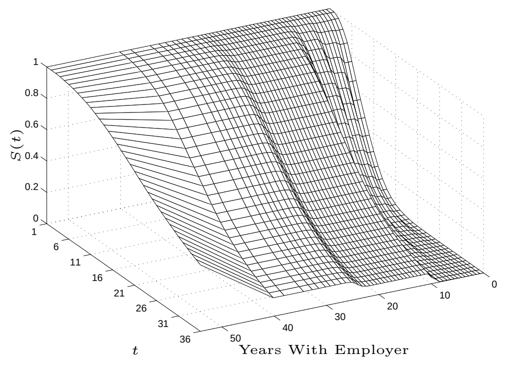
(a) Age



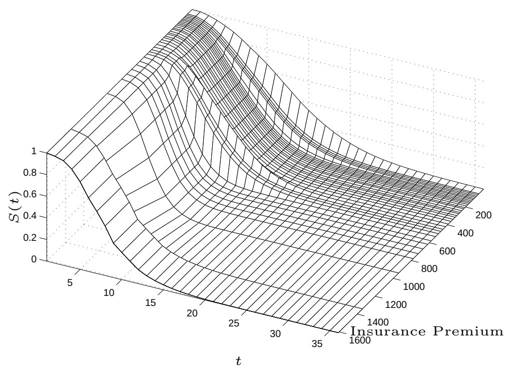
(b) Amount



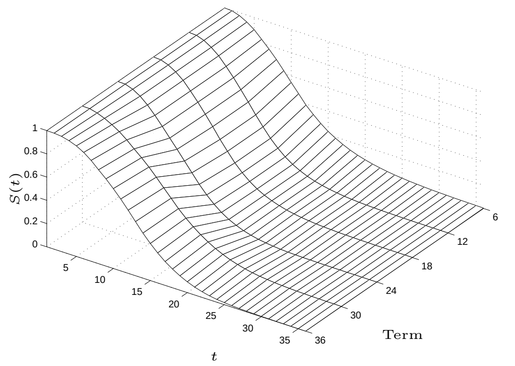
(c) Years at Current Address



(d) Years with Employer



(e) Insurance Premium



(f) Term

Figure 6.7: Evolution of neural network survival distributions with respect to inputs for predicting default.

6.6 Conclusions

In this chapter, we studied the use of survival analysis methods for credit scoring. Traditionally, credit scoring aimed at distinguishing good customers from bad customers. However, knowledge of the timing of when customers default or pay off early has become more and more interesting for e.g. calculating the profit over a customer's lifetime. In the literature, this problem has been typically tackled using statistical survival analysis methods such as the proportional hazards method. However, these models suffer from a number of drawbacks: the functional form of the inputs remains linear or some mild extension thereof, non-linearity and interaction effects have to be explicitly and rather arbitrarily modeled by the statistician, and in the standard proportional hazards model, the baseline hazard function is assumed to be uniform and proportional across the entire population.

Multilayer perceptron neural networks have been proposed as a solution to the above problems. Being non-linear, universal approximators, neural networks may be a very attractive alternative for survival analysis modeling. Many approaches have already been suggested in the literature to use neural networks for survival analysis. Each of them differs in the way of modeling censored observations, time-varying inputs and scalability. In this chapter, we provided a literature overview of neural network approaches that have been (recently) suggested for survival analysis modeling. Only a few approaches were found to be scalable and generated monotonically decreasing survival curves.

In the empirical part of this chapter, we compared the performance of the proportional hazards model with that of a neural network based survival analysis model on a data set of 15000 observations. We investigated when customers default as well as when they pay off their loan early. It was found that, for early repayment, the suggested neural network approach outperformed the proportional hazards model. For predicting default, the superiority of the neural network model was somewhat less pronounced.

Chapter 7

Conclusions

In this PhD dissertation, we discussed how machine learning techniques may be adopted to develop intelligent systems for credit scoring. In what follows, we will provide a chapter-by-chapter overview of the thesis and indicate the major contributions. Furthermore, we will also outline some issues for further research.

7.1 Thesis Summary and Principal Conclusions

The major goal of credit scoring is to distinguish between good and bad payers by using a model which was estimated based on the repayment behavior of a set of applicants from the past. This problem statement essentially reduces to a binary classification problem. Many techniques have been suggested to tackle this. In chapter 2, we started with providing an overview of a selection of classification techniques. We discussed statistical classifiers (logistic regression, linear, Fisher and quadratic discriminant analysis), linear programming, naive Bayes, tree augmented naive Bayes, C4.5, C4.5rules, k-nearest neighbor, neural networks and (least squares) support vector machine classifiers. This was followed with an overview of some issues relating to the practical implementation of a classifier. Topics that were addressed are: how to split up the data in order to assess the performance of a classifier, and how to reduce the number of inputs used by a classifier in order to make the classification model more simple and concise. We then conducted a first benchmarking study in order to validate the performance of some of the classification techniques discussed on 10 publicly available data sets. Advanced parameter tuning methods were adopted and a rigorous statistical setup was employed. A second benchmarking study revealed the performance of linear and quadratic discriminant analysis, logistic regression, and least squares support vector machines on the highly non-linear checkerboard and 2-spiral classification problems. From the first study, it was concluded that the non-linear RBF (least squares) support vector machine classifiers consistently yielded a very good performance on all data sets. However, the more simple, linear classifiers such as linear discriminant analysis and logistic regression also give satisfactory performance on most of the data sets considered. In the second study, the superiority of the non-linear least squares support vector machine classifier was more pronounced than in the first study. This allowed us to conclude that many real-life classification data sets are very good separable using simple classification techniques although non-linear classifiers may provide additional performance benefits which may be very important in a data mining context. A criticism concerning both studies was that they only considered the classification accuracy whereas misclassification costs and class distributions are also related to classification performance. It was then argued that the area under the receiver operating characteristic curve is a performance measure independent of misclassification costs or class distribution. The chapter was concluded with a discussion of the McNemar and DeLong, DeLong and Clarke-Pearson test to compare the classification accuracy and area under the receiver operating characteristic curve, respectively.

In chapter 3, we studied the topic of building scorecards for credit scoring using the classification techniques discussed in chapter 2. The chapter started with a brief overview of the basic problem statement of credit scoring. Next, we discussed the problem of reject inference and provided a literature overview on the use of machine learning techniques for credit scoring. In the experimental part, we used 8 real-life credit scoring data sets. Two data sets originated from Benelux financial institutions, four data sets from U.K. financial institutions and two were publicly available data sets. We reported both the classification accuracy and the area under the receiver operating characteristic curve. For the former, we experimented with various cut-off setting schemes, e.g. a cut-off of 0.5, a cut-off assuming equal sample proportions, and cut-offs assuming marginal good-bad rates around 5:1 and 3:1, respectively. Again a rigorous statistical setup was employed. It was concluded that the least squares support vector machine classifier using a radial basis function kernel and the neural network consistently performed very good on all data sets. However, the more simple, linear techniques such as logistic regression and linear discriminant analysis also yielded good classification performances which clearly indicates that most credit scoring data sets are only weakly non-linear. It was also remarked that it is our firm belief that the best way to augment the performance of a scorecard is to look for better discriminatory predictors. The development of credit reference agencies or credit bureaus may play a pivotal role in this context.

It was extensively argued in this dissertation that accuracy and comprehensibility are two key properties of a successful credit scoring system. A credit scoring expert will typically have low confidence and trust in using a mathematically complex, highly parameterized scorecard because of its low comprehensive value. Hence, such black box decision models are likely not to be successfully integrated and deployed into the daily credit decision environment. Starting from the observation that neural networks achieved very good classification performance on all credit scoring data sets, we investigated in chapter 4 how their black box property can be solved using crisp rule extraction techniques. The following rule extraction techniques were studied: Neurolinear, Neurorule and Trepan. Neurolinear and Neurorule are both decompositional rule extraction techniques extracting oblique and propositional rules, respectively. Trepan is a pedagogical tree extraction algorithm generating trees with M-of-N type of splits. The experiments were conducted on two real-life Benelux data sets and a publicly available credit scoring data set. Both the continuous and the discretized versions were analyzed. We reported the classification accuracy, the complexity (i.e. the number of rules, or number of leave nodes and total number of nodes for the trees), and the fidelity which is the percentage of observations which the extracted rule set or tree classifies in the same way as the neural network. It was found that especially Neurorule extracted concise, comprehensible rule sets with a high classification accuracy. In a final step, we used decision tables to represent the extracted rules and trees in an efficient and user-friendly manner. Hence, neural network rule extraction and decision tables are two powerful management science tools that allow to build accurate and comprehensible credit decision support systems.

The neural network rule extraction techniques studied in chapter 4 extract crisp rules where the antecedents are either true or false. In chapter 5, we investigated the use of fuzzy rule extraction techniques for building intelligent credit scoring systems. Fuzzy rules use fuzzy sets in the antecedents to express vague, linguistic concepts which are believed to be more close to human reasoning. A distinction was made between approximate and descriptive fuzzy rules. Approximate fuzzy rules use their own definition of membership functions whereas descriptive fuzzy rules refer to a commonly defined set of membership functions. Descriptive fuzzy rules are easier to understand and comprehend than their approximate counterparts. Many learning paradigms have been suggested to extract fuzzy rules from a set of data. In chapter 5, we discussed the use of boosted evolutionary algorithms to extract descriptive and approximate fuzzy rules and contrasted this with Nefclass, a neurofuzzy classifier generating descriptive fuzzy rules. The experiments were conducted on 4 credit scoring data sets, 2 publicly available data sets and 1 artificially generated data set. Comparisons were made with a selection of well-known classification algorithms. It was concluded that the evolutionary fuzzy rule learners compare favorably to the other algorithms in terms of classification accuracy. The descriptive and approximate fuzzy rules, extracted by the evolutionary classifiers, yielded more or less the same accuracy across all data sets which indicates that the boosting scheme compensated for the expressive weaknesses of the individual classification rules. However, it was noted that the weighting and voting scheme complicates the interpretation.

In all previous chapters, the purpose was to distinguish good payers from bad payers using their application characteristics. However, knowing when customers default is becoming more and more important since it can provide the financial institution with the ability to perform profit scoring. In chapter 6, we discussed the use of survival analysis methods to analyze when customers default. We contrasted the use of the proportional hazards model, which is the most well-known statistical survival analysis model, with the use of neural network survival analysis models. The advantage of using the latter is that neural networks can model non-linearities, complex interactions and do not rely on a proportional hazards assumption. After an extensive literature overview, both the proportional hazards model and a neural network model were compared on a real-life credit scoring data set. We investigated both methods for predicting early repayment and default. It was concluded that, for early repayment, the suggested neural network approach outperformed the proportional hazards model. For predicting default, the superiority of the neural network was somewhat less pronounced.

7.2 Issues for Further Research

Starting from the conclusions and findings of this dissertation, many challenging issues for further research can be identified.

7.2.1 The Knowledge Fusion problem

In this dissertation, our major goal was to develop scorecards by means of machine learning algorithms trained using a set of given observations representing the repayment behavior of clients in the past. However, it needs to be noted that although machine learning algorithms are very powerful, they generally rely on modeling repeated patterns or correlations which occur in the data. It may well occur that observations, which are very evident to classify by the domain expert, do not appear frequently enough in the data in order to be appropriately modeled by a machine learning algorithm. Hence, the intervention and interpretation of the domain expert still remains crucial. Especially in a credit scoring context, financial institutions typically already apply certain credit policies and rules based on the experience and expertise of one or more credit scoring experts. This prescreening procedure forms the basis of the well-known reject inference problem discussed in chapter 3. Essentially, one is confronted with two kinds of knowledge: knowledge extracted from data and knowledge representing the experience of the domain expert(s). Ideally, one should try to elicit the latter and merge it with the former into one coherent credit scoring decision support system (see Figure 7.1). However, this knowledge fusion exercise is very challenging from an academic viewpoint and forms an interesting avenue for future research.

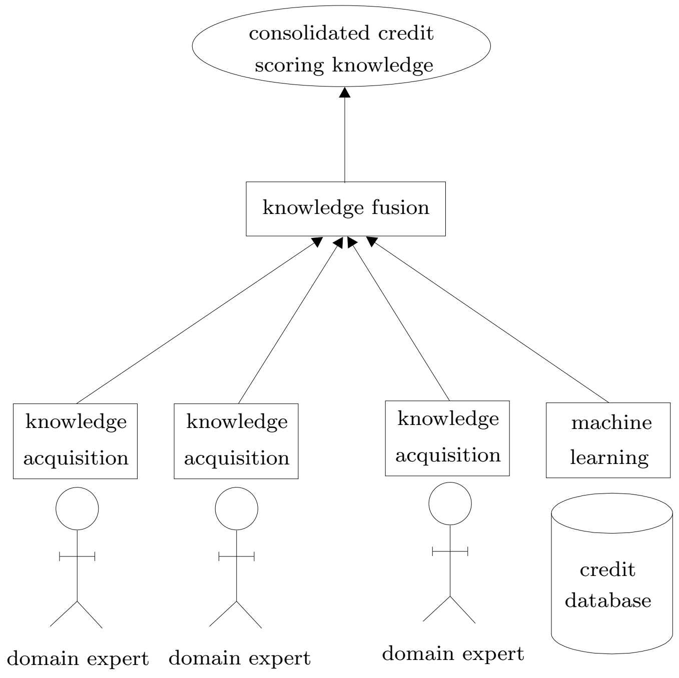


Figure 7.1: The Knowledge Fusion process.

7.2.2 Extensions to Indirect Credit Scoring

In this dissertation, we tackled the credit scoring problem as a classification problem. For most of the credit scoring data sets considered, a bad customer was defined as someone who has missed three consecutive months of payments. In indirect credit scoring, one first tries to predict the number of months in payment arrears (or other criteria e.g., balance excess, turnover, ...) and then uses a deterministic model to label customers as good or bad. The advantage of indirect credit scoring is that it lends itself very well to using alternative definitions of good/bad, i.e. one only needs to reformulate the deterministic model instead of having to re-estimate the prediction model.

It would be interesting to investigate how our findings and results can be translated from a direct credit scoring context to an indirect credit scoring context. The classification exercise then essentially becomes a regression exercise with a continuous dependent variable. This would mean that we would have to include other techniques in our benchmarking experiment. E.g., for the statistical part, one should include Ordinary Least Squares (OLS) regression whereas for the decision tree part, the CART (Classification And Regression Trees) method may be interesting. One could also use neural network rule extraction in order to have more comprehensible models. Techniques that could be considered in this context are REFANN (Rule Extraction from Function Approximating Neural Networks) [207] for crisp rule extraction and ANFIS (Adaptive Network-Based Fuzzy Inference System) for fuzzy rule extraction [127].

7.2.3 Behavioral Credit Scoring

It would be interesting to see how the ideas presented in this dissertation in an application credit scoring setting, can be translated to a behavioral credit scoring setting. Instead of only looking at the application characteristics, behavioral models also try to capture some of the dynamics of the customer behavior by monitoring their financial and/or socio-demographic status. The models are then dynamically updated at regular time intervals and new scores can be computed indicating the revised future risk profile of the customer. A challenging issue for further research is how to adequately incorporate these dynamics into the credit models by, e.g., building neural network survival analysis models having time-varying inputs.

7.2.4 Extensions to other Contexts and Problem Domains

The ideas presented in this dissertation may be easily transferred and applied in other relevant (classification) settings. An example is the closely related problem of bankruptcy prediction where the aim is to distinguish between solvent and non-solvent firms using a set of characteristics describing their financial status. Also, in a marketing context, problems such as churn prediction and customer retention are very well suited to be investigated. In a medical context, it would be interesting to develop medical decision support systems using the ideas and concepts presented in this dissertation.

Appendix A

Tables Accompanying Benchmarking Study 1 of Chapter 2

LS-SVM	`acr`	`bld`	`gcr`	`hea`	`ion`	`pid`	`snr`	`ttt`	`wbc`	`adu`
N_{CV}	460	230	666	180	234	512	138	638	455	33000
n	14	6	20	13	33	8	60	9	9	14
RBF	0.86	0.75	0.83	0.86	1.00	0.78	0.94	1.00	0.98	0.85
Lin	0.87	0.69	0.75	0.85	0.90	0.78	0.88	0.66	0.96	0.82
Pol $d = 2$	0.89	0.75	0.82	0.87	0.99	0.79	0.98	0.98	0.98	0.85
Pol $d = 3$	0.88	0.76	0.91	0.88	0.99	0.78	0.98	1.00	0.98	0.85
Pol $d = 4$	0.86	0.76	0.93	0.89	0.93	0.78	0.96	1.00	0.98	0.85
Pol $d = 5$	0.89	0.76	0.94	0.85	0.90	0.78	0.99	1.00	0.97	0.85
Pol $d = 6$	0.88	0.75	0.91	0.86	0.94	0.79	0.99	1.00	0.98	0.85
Pol $d = 7$	0.89	0.76	0.94	0.85	0.95	0.80	0.96	1.00	0.98	0.85
Pol $d = 8$	0.89	0.76	0.91	0.85	0.98	0.81	0.99	1.00	0.98	0.85
Pol $d = 9$	0.88	0.76	0.93	0.86	0.93	0.78	0.98	1.00	0.98	0.85
Pol $d = 10$	0.88	0.78	0.95	0.85	0.99	0.78	0.98	1.00	0.98	0.85

Table A.1: Training set performance of LS-SVMs on 10 binary data sets.

LS-SVM	`acr`	`bld`	`gcr`	`hea`	`ion`	`pid`	`snr`	`ttt`	`wbc`	`adu`
RBF: σ	22.75	41.25	31.25	5.69	3.30	240.00	33.00	2.93	6.97	10.0
RBF: $\log_{10}(\gamma)$	0.09	3.01	2.43	-0.76	0.63	3.04	0.86	2.20	-0.66	1.02
Lin: $\log_{10}(\gamma)$	-2.29	0.14	-1.82	-1.51	-1.99	-0.21	-2.26	-2.68	-1.53	-0.82
Pol: d	5	3	6	2	2	3	2	4	3	3
Pol: c	5.61	15.30	5.86	1.80	7.18	42.42	3.87	3.00	5.25	5.61
Pol: $\log_{10}(\gamma)$	-1.66	1.26	-1.20	-2.51	1.38	1.29	-1.27	0.45	-0.91	0.02

Table A.2: Optimized hyperparameter values of the LS-SVMs with RBF, linear and polynomial kernels for the UCI classification data sets.

SVM	`acr`	`bld`	`gcr`	`hea`	`ion`	`pid`	`snr`	`ttt`	`wbc`	`adu`
RBF: σ	12.43	9.0	55.0	7.15	3.30	15.50	5.09	9.00	19.5	8.00
RBF: $\log_{10}(C)$	2.09	1.64	3.68	-0.51	0.51	0.04	1.70	-0.41	1.86	0.70
Lin: $\log_{10}(C)$	-2.43	1.57	1.45	1.32	1.20	-2.08	-1.05	-4.25	-2.12	-2.30

Table A.3: Optimized hyperparameter values of the SVMs with RBF and linear kernel for the UCI classification data sets.

Appendix B

Multiclass Benchmarking Study

The benchmarking study reported in section 2.4 of chapter 2 was also extended to a multiclass setting. All data sets have been obtained from the publicly accessible UCI repository [27] at `http://kdd.ics.uci.edu/`, except for the US postal service data set which was retrieved from `http://www.kernel-machines.org/`. Their characteristics are displayed in Table B.1. Note that in Table B.1, N_{CV} stands for the number of data points used in the cross-validation based tuning procedure, N_{test} for the number of observations in the test set (see subsection 2.4.2) and N for the total data set size. The number of numerical and categorical attributes is denoted by n_{num} and n_{cat} respectively, n is the total number of attributes.

For the multiclass SVM and LS-SVM classifiers, we experimented with different types of output coding schemes. In minimum output coding (MOC), one uses

	`bal`	`cmc`	`ims`	`iri`	`led`	`thy`	`usp`	`veh`	`wav`	`win`
N_{CV}	416	982	1540	100	2000	4800	6000	564	2400	118
N_{test}	209	491	770	50	1000	2400	3298	282	1200	60
N	625	1473	2310	150	3000	7200	9298	846	3600	178
n_{num}	4	2	18	4	0	6	256	18	19	13
n_{cat}	0	7	0	0	7	15	0	0	0	0
n	4	9	18	4	7	21	256	18	19	13
M	3	3	7	3	10	3	10	4	3	3
L_{MOC}	2	2	3	2	4	2	4	2	2	2
L_{1vs1}	3	3	21	3	45	3	45	6	2	3

Table B.1: Characteristics of the multiclass classification data sets.

$\lceil \frac{\ln M}{\ln 2} \rceil$ binary classifiers, where M is the number of classes and $\lceil \cdot \rceil$ rounds toward $+\infty$. In One-versus-One output coding (1vs1), $M(M-1)/2$ binary classifiers are constructed to discriminate between each pair of two classes. A new observation is then assigned to the output code with minimal Hamming distance (see [240] for more details). The M row in Table B.1 denotes the number of classes for each data set, encoded by L_{MOC} and L_{1vs1} bits for MOC and 1vs1 output coding, respectively.

Table B.2 reports the performance of the multiclass classifiers on the 10 data sets. It has the same setup as Table 2.4 from chapter 2. The same kernel types as for the binary data sets were considered: RBF kernels, linear (Lin) and polynomial (Pol) kernels with degrees $d = 2, \ldots, 10$. Both the performance of LS-SVM and LS-SVM with Fisher targets (LS-SVM$_{\text{F}}$) classifiers are reported. The MOC and 1vs1 output coding were also applied to the SVM classifiers with linear and RBF kernels.

The experimental setup outlined in sections 2.4.2 and 2.4.3 is used: each binary classifier of the multiclass (LS-)SVM is designed on the first 2/3 of the data using 10-fold cross-validation, while the remaining 1/3 are put aside for testing. The selected regularization and kernel parameters were then fixed and 10 randomizations were conducted for each data set. The average test set accuracies of the different LS-SVM and LS-SVM$_{\text{F}}$ classifiers, with RBF, Lin and Pol kernel ($d = 2, \ldots, 10$) and using MOC and 1vs1 output coding, are reported in Table B.2. The test set accuracies of the reference algorithms on the same randomizations are also reported, where we remark that for the `usp` data set the memory requirements for logit were too high. Instead, we tabulated the performance of logit for this single case. The same statistical tests as in section 2.4.3 were used to compare the performance of the different classifiers.

The use of QDA yields the best average test set accuracy on two data sets, while LS-SVMs with 1vs1 coding using a RBF and Lin kernel and LS-SVM$_{\text{F}}$ with Lin kernel each yield the best performance on one data set. SVMs with RBF kernel with MOC and 1vs1 coding yield the best performance on one data set each. Also C4.5, logit and IB1 each achieve one time the best performance. The use of 1vs1 coding generally results into a better classification accuracy. Averaging over all 10 multiclass data sets, the LS-SVM classifier with RBF kernel and 1vs1 output coding achieves the best average accuracy (AA) and average ranking, while its performance is only on three domains significantly worse at the 1% level than the best algorithm. This performance is not significantly different from the SVM with RBF kernel and 1vs1 output coding. Summarizing the different significance tests, RBF LS-SVM (MOC), Pol LS-SVM (MOC), Lin LS-SVM (1vs1), Lin LS-SVM$_{\text{F}}$ (1vs1), Pol LS-SVM (1vs1), RBF SVM (MOC), RBF SVM (1vs1), Lin SVM (1vs1), LDA, QDA, C4.5, IB1 and IB10 perform not significantly different at the 5% level. While NB$_{\text{k}}$ performed well on the binary data sets, its average accuracy in the multiclass case is never comparable at the 5% level for all three tests. The results of Table B.2 illustrate that the SVM and LS-SVM classifier with RBF kernel using 1vs1 output coding consistently yield very good test set accuracies on the multiclass data sets.

	bal	cmc	ims	iri	led	thy	usp	veh	wav	win	AA	AR	P_{ST}
N_{test}	209	491	770	50	1000	2400	3298	282	1200	60			
n	4	9	18	4	7	21	256	18	19	13			
RBF LS-SVM (MOC)	*92.7*(1.0)	**54.1**(1.8)	*95.5*(0.6)	*96.6*(2.8)	*70.8*(1.4)	*96.6*(0.4)	*95.3*(0.5)	81.9(2.6)	**99.8**(0.2)	**98.7**(1.3)	**88.2**	**7.1**	**0.344**
RBF LS-SVM$_F$ (MOC)	*86.8*(2.4)	*43.5*(2.6)	*69.6*(3.2)	**98.4**(2.1)	*36.1*(2.4)	*22.0*(4.7)	*86.5*(1.0)	*66.5*(6.1)	*99.5*(0.2)	*93.2*(3.4)	*70.2*	17.8	**0.109**
Lin LS-SVM (MOC)	*90.4*(0.8)	*46.9*(3.0)	*72.1*(1.2)	*89.6*(5.6)	*52.1*(2.2)	*93.2*(0.6)	*76.5*(0.6)	*69.4*(2.3)	*90.4*(1.1)	**97.3**(2.0)	*77.8*	*17.8*	*0.002*
Lin LS-SVM$_F$ (MOC)	*86.6*(1.7)	*42.7*(2.0)	*69.8*(1.2)	*77.0*(3.8)	*35.1*(2.6)	*54.1*(1.3)	*58.2*(0.9)	*69.1*(2.0)	*55.7*(1.3)	*85.5*(5.1)	*63.4*	*22.4*	*0.002*
Pol LS-SVM (MOC)	*94.0*(0.8)	53.5(2.3)	*87.2*(2.6)	**96.4**(3.7)	*70.9*(1.5)	*94.7*(0.2)	*95.0*(0.8)	*81.8*(1.2)	*99.6*(0.3)	**97.8**(1.9)	87.1	**9.8**	**0.109**
Pol LS-SVM$_F$ (MOC)	*93.2*(1.9)	*47.4*(1.6)	*86.2*(3.2)	96.0(3.7)	*67.7*(0.8)	*69.9*(2.8)	*87.2*(0.9)	*81.9*(1.3)	*96.1*(0.7)	*92.2*(3.2)	*81.8*	*15.7*	*0.002*
RBF LS-SVM (1vs1)	*94.2*(2.2)	**55.7**(2.2)	**96.5**(0.5)	**97.6**(2.3)	**74.1**(1.3)	*96.8*(0.3)	94.8(2.5)	83.6(1.3)	*99.3*(0.4)	**98.2**(1.8)	**89.1**	**5.9**	**1.000**
RBF LS-SVM$_F$ (1vs1)	*71.4*(15.5)	*42.7*(3.7)	*46.2*(6.5)	*79.8*(10.3)	*58.9*(8.5)	*92.6*(0.2)	*30.7*(2.4)	*24.9*(2.5)	*97.3*(1.7)	*67.3*(14.6)	*61.2*	*22.3*	*0.002*
Lin LS-SVM (1vs1)	*87.8*(2.2)	*50.8*(2.4)	*93.4*(1.0)	**98.4**(1.8)	**74.5**(1.0)	*93.2*(0.3)	*95.4*(0.3)	*79.8*(2.1)	*97.6*(0.9)	**98.3**(2.5)	**86.9**	**9.7**	**0.754**
Lin LS-SVM$_F$ (1vs1)	*87.7*(1.8)	*49.6*(1.8)	*93.4*(0.9)	**98.6**(1.3)	**74.5**(1.0)	*74.9*(0.8)	*95.3*(0.3)	*79.8*(2.2)	*98.2*(0.6)	**97.7**(1.8)	85.0	**11.1**	**0.344**
Pol LS-SVM (1vs1)	*95.4*(1.0)	*53.2*(2.2)	*95.2*(0.6)	96.8(2.3)	72.8(2.6)	88.8(14.6)	**96.0**(2.1)	82.8(1.8)	*99.0*(0.4)	**99.0**(1.4)	**87.9**	**8.9**	**0.344**
Pol LS-SVM$_F$ (1vs1)	*56.5*(16.7)	*41.8*(1.8)	*30.1*(3.8)	*71.4*(12.4)	*32.6*(10.9)	*92.6*(0.7)	95.8(1.7)	*20.3*(6.7)	*77.5*(4.9)	*82.3*(12.2)	*60.1*	*21.9*	0.021
RBF SVM (MOC)	**99.2**(0.5)	*51.0*(1.4)	*94.9*(0.9)	96.6(3.4)	*69.9*(1.0)	*96.6*(0.2)	*95.5*(0.4)	*77.6*(1.7)	**99.7**(0.1)	**97.8**(2.1)	**87.9**	**8.6**	**0.344**
Lin SVM (MOC)	*98.3*(1.2)	*45.8*(1.6)	*74.1*(1.4)	**95.0**(10.5)	*50.9*(3.2)	*92.5*(0.3)	*81.9*(0.3)	*70.3*(2.5)	*99.2*(0.2)	97.3(2.6)	80.5	16.1	0.021
RBF SVM (1vs1)	*98.3*(1.2)	**54.7**(2.4)	*96.0*(0.4)	97.0(3.0)	*64.6*(5.6)	*98.3*(0.3)	**97.2**(0.2)	83.8(1.6)	*99.6*(0.2)	**96.8**(5.7)	**88.6**	**6.5**	**1.000**
Lin SVM (1vs1)	*91.0*(2.3)	*50.8*(1.6)	*95.2*(0.7)	**98.0**(1.9)	**74.4**(1.2)	*97.1*(0.3)	*95.1*(0.3)	*78.1*(2.4)	*99.6*(0.2)	**98.3**(3.1)	**87.8**	**7.3**	**0.754**
LDA	*86.9*(2.1)	*51.8*(2.2)	*91.2*(1.1)	**98.6**(1.0)	73.7(0.8)	*93.7*(0.3)	*91.5*(0.5)	*77.4*(2.7)	*94.6*(1.2)	**98.7**(1.5)	85.8	**11.0**	**0.109**
QDA	*90.5*(1.1)	*50.6*(2.1)	*81.8*(9.6)	**98.2**(1.8)	**73.6**(1.1)	*93.4*(0.3)	*74.7*(0.7)	**84.8**(1.5)	*60.9*(9.5)	**99.2**(1.2)	**80.8**	**11.8**	**0.344**
Logit	*88.5*(2.0)	*51.6*(2.4)	*95.4*(0.6)	**97.0**(3.9)	**73.9**(1.0)	*95.8*(0.5)	*91.5*(0.5)	*78.3*(2.3)	**99.9**(0.1)	*95.0*(3.2)	*86.7*	**9.8**	0.021
C4.5	*66.0*(3.6)	*50.9*(1.7)	96.1(0.7)	96.0(3.1)	73.6(1.3)	**99.7**(0.1)	*88.7*(0.3)	*71.1*(2.6)	**99.8**(0.1)	*87.0*(5.0)	82.9	**11.8**	**0.109**
oneR	*59.5*(3.1)	*43.2*(3.5)	*62.9*(2.4)	*95.2*(2.5)	*17.8*(0.8)	*96.3*(0.5)	*32.9*(1.1)	*52.9*(1.9)	*67.4*(1.1)	*76.2*(4.6)	*60.4*	*21.6*	*0.002*
IB1	*81.5*(2.7)	*43.3*(1.1)	**96.8**(0.6)	*95.6*(3.6)	**74.0**(1.3)	*92.2*(0.4)	*97.0*(0.2)	*70.1*(2.9)	*99.7*(0.1)	*95.2*(2.0)	**84.5**	**12.9**	**0.344**
IB10	*83.6*(2.3)	*44.3*(2.4)	*94.3*(0.7)	*97.2*(1.9)	**74.2**(1.3)	*93.7*(0.3)	*96.1*(0.3)	*67.1*(2.1)	*99.4*(0.1)	*96.2*(1.9)	84.6	**12.4**	**0.344**
NB$_k$	*89.9*(2.0)	*51.2*(2.3)	*84.9*(1.4)	**97.0**(2.5)	**74.0**(1.2)	*96.4*(0.2)	*79.3*(0.9)	*60.0*(2.3)	*99.5*(0.1)	**97.7**(1.6)	*83.0*	12.2	0.021
NB$_n$	*89.9*(2.0)	*48.9*(1.8)	*80.1*(1.0)	**97.2**(2.7)	**74.0**(1.2)	*95.5*(0.4)	*78.2*(0.6)	*44.9*(2.8)	*99.5*(0.1)	97.5(1.8)	*80.6*	13.6	0.021
Maj. Rule	*48.7*(2.3)	*43.2*(1.8)	*15.5*(0.6)	*38.6*(2.8)	*11.4*(0.0)	*92.5*(0.3)	*16.8*(0.4)	*27.7*(1.5)	*34.2*(0.8)	*39.7*(2.8)	*36.8*	*24.8*	*0.002*

Table B.2: The multiclass benchmarking study.

Appendix C

Attributes for German credit

Nr	**Name**	**Type**	**Explanation**
1	Checking account	nominal	1: < 0 DM; 2: ≥ 0 and < 200 DM; 3: ≥ 200 DM/salary assignments for at least one year; 4: no checking account
2	Term	continuous	
3	Credit history	nominal	0: no credits taken/all credits paid back duly; 1: all credits at this bank paid back duly; 2: existing credits paid back duly till now; 3: delay in paying off in the past; 4: critical account/ other credits (not at this bank)
4	Purpose	nominal	0: car (new); 1: car (old); 2: furniture/equipment; 3: radio/ television; 4: domestic appliances; 5: repairs; 6: education; 7: vacation; 8: retraining; 9: business; 10: others
5	Credit amount	continuous	
6	Savings account	nominal	1: < 100 DM; 2: ≥ 100 DM and < 500 DM; 3: ≥ 500 and < 1000 DM; 4: ≥ 1000 DM; 5: unknown/no savings account
7	Present employment since	nominal	1: unemployed; 2: < 1 year; 3: ≥ 1 year and < 4 years; 4: ≥ 4 and < 7 years; 5: ≥ 7 years

8	Installment rate (% of disposable income)	continuous	
9	Personal status and sex	nominal	1: male,divorced/separated; 2: female, divorced/separated/ married; 3: male, single; 4: male, married/widowed; 5: female,single
10	Other parties	nominal	1: none; 2: co-applicant; 3: guarantor
11	Present residence since	continuous	
12	Property	nominal	1: real estate; 2: if not 1: building society savings agreement/life insurance; 3: if not 1/2: car or other; 4: unknown/no property
13	Age	continuous	
14	Other installment plans	nominal	1: bank; 2: stores 3: none
15	Housing	nominal	1: rent; 2: own 3: for free
16	Number of existing credits at this bank	continuous	
17	Job	nominal	1: unemployed/unskilled-non-resident; 2: unskilled-resident; 3: skilled employee/official; 4: management/ self employed/ highly qualified employee/officer
18	Number of dependents	continuous	
19	Telephone	nominal	1: none; 2: yes, registered under the customer name
20	Foreign worker	nominal	1: yes; 2: no

Table C.1: Attributes for the German credit data set.

Appendix D

Attributes for Bene1

Nr	Name	Type
1	Identification number	continuous
2	Amount of loan	continuous
3	Amount on purchase invoice	continuous
4	Percentage of financial burden	continuous
5	Term	continuous
6	Personal loan	nominal
7	Purpose	nominal
8	Private or professional loan	nominal
9	Monthly payment	continuous
10	Savings account	continuous
11	Other loan expenses	continuous
12	Income	continuous
13	Profession	nominal
14	Number of years employed	continuous
15	Number of years in Belgium	continuous
16	Age	continuous
17	Applicant Type	nominal
18	Nationality	nominal
19	Marital status	nominal
20	Number of years since last house move	continuous
21	Code of regular saver	nominal
22	Property	nominal
23	Existing credit info	nominal
24	Number of years client	continuous
25	Number of years since last loan	continuous
26	Number of checking accounts	continuous

27	Number of term accounts	continuous
28	Number of mortgages	continuous
29	Number of dependents	continuous
30	Pawn	nominal
31	Economical sector	nominal
32	Employment status	nominal
33	Title/salutation	nominal

Table D.1: Attributes for the Bene1 data set.

List of Figures

I Het reject inference probleem. xiii

II Neuraal netwerk voor het voorspellen van kredietwaardigheid. . . . xvi

III Als-dan'-regels geëxtraheerd uit het neuraal netwerk van Figuur II. xvii

IV Het gebruik van vage regels voor krediettoekening. xvii

V Beslissingstabel voor de regels van Figuur III. xviii

VI Het *knowledge fusion* proces. xix

1.1 The Knowledge Discovery in Data process. 2

2.1 Quadratic versus Linear Discriminant analysis. 13

2.2 An example Bayesian network classifier [73]. 16

2.3 Example of a Bayesian network classifier for marketing [14]. 16

2.4 Example of a Bayesian network classifier for credit scoring [8]. . . . 17

2.5 The naive Bayes classifier. 18

2.6 The tree augmented naive Bayes classifier. 19

2.7 Example Decision Tree [163]. 20

2.8 The Entropy measure. 20

2.9 The 5-nearest neighbor classifier [67]. 23

2.10 Architecture of a Multilayer Perceptron with one hidden layer. . . 24

2.11 Neural network transfer functions. 25

2.12 Illustration of SVM optimization of the margin in the feature space. 27

2.13 10-fold cross-validation. 31

2.14 Input search space. 32

2.15 Cross-validation (CV10) classification accuracy on the `ion` data set. 36

2.16 The checkerboard and 2-spiral classification problem. 42

2.17 LS-SVM generalisation behaviour on the checkerboard and 2-spiral classification problem. 43

2.18 The receiver operating characteristic curve (ROC). 47

3.1 Direct versus indirect credit scoring. 55

3.2 The reject inference problem. 56

4.1 Oblique rules versus propositional rules. 78

4.2 Example network used by Neurorule and Neurolinear for rule extraction. 83

4.3 Oblique rules extracted by Neurolinear for German credit. 91

4.4 Oblique rules extracted by Neurolinear for Bene1. 92

4.5 Neural network trained and pruned for Bene1. 93

4.6 Rules Extracted by Neurorule for German credit. 96

4.7 Rules Extracted by Neurorule for Bene1. 96

4.8 Tree extracted by Trepan for German credit. 97

4.9 Tree extracted by Trepan for Bene1. 97

4.10 DT quadrants. 98

4.11 Minimizing the number of columns of a DT. 99

4.12 Example of an unordered DT. 99

4.13 DT size distribution [9]. 101

4.14 Decision table for the rules extracted by Neurorule on German credit.103

4.15 Decision table for the rules extracted by Neurorule on Bene1. . . . 103

4.16 Example consultation session in PROLOGA. 104

4.17 Classifying an applicant in PROLOGA. 105

5.1 Fuzzy sets and corresponding membership functions for age. 111

5.2 Approximate fuzzy rules versus Descriptive fuzzy rules. 112

5.3 Outline of an evolutionary algorithm [36, 38]. 113

5.4 Architecture of a boosted evolutionary classifier. 115

5.5 Coding of trapezoidal fuzzy sets. 116

5.6 Example Nefclass network. 121

5.7 Evolution of the training and test set classification accuracy with the number of fuzzy rules generated for the evolutionary fuzzy classifiers on the Bene2 data set. 126

5.8 Approximate fuzzy rules for the Bene2 data set. The weights **w** correspond to the $\log(1/\beta_t)$ factors (see equation 5.17). 128

5.9 Descriptive fuzzy rules for the Bene2 data set using 5 fuzzy sets. The weights **w** correspond to the $\log(1/\beta_t)$ factors (see equation 5.17). 129

6.1 Censoring. 132

6.2 Survival and hazard functions for an exponentially distributed event time distribution $f(t)$ with $\lambda = 0.75$. 135

6.3 An example of a modular neural network for survival analysis whereby the output of the networks predicting $S(t_i)$ and $S(t_j)$ are used as additional inputs for the network predicting $S(t_k)$. 142

6.4 Kaplan Meier curves for default and early repayment. 150

6.5 Encoding of the neural network outputs for survival analysis. . . . 152

6.6 Evolution of neural network survival distributions with respect to inputs for predicting early repayment. 157

6.7 Evolution of neural network survival distributions with respect to inputs for predicting default. 161

7.1 The Knowledge Fusion process. 167

List of Tables

I Voorspellende versus beschrijvende data mining. xiv

1.1 Types of data mining. 3

1.2 Example machine learning algorithms. 3

2.1 Characteristics of the binary classification UCI data sets. 34

2.2 Validation set performance of LS-SVMs on 10 binary data sets. . . 38

2.3 Test set performance of LS-SVMs on 10 binary data sets. 39

2.4 Comparison of the 10 times randomized test set performance of all classifiers. 41

2.5 LDA, QDA, logit and LS-SVM classification accuracy on the checkerboard and 2-spiral classification problem. 42

2.6 Economics resulting from performance differences among response models. 44

2.7 The confusion matrix for binary classification. 45

2.8 Contingency table for McNemar test. 48

3.1 Characteristics of credit scoring data sets. 61

3.2 The marginal good-bad rate. 64

3.3 Test set classification accuracy on credit scoring data sets assuming a cut-off of 0.5. 67

3.4 Test set classification accuracy on credit scoring data sets assuming equal sample proportions. 68

3.5 Test set classification accuracy on credit scoring data sets assuming a marginal good-bad rate around 5:1. 69

3.6 Test set classification accuracy on credit scoring data sets assuming a marginal good-bad rate around 3:1. 70

3.7 Test set AUC on credit scoring data sets. 71

4.1 The XOR classification problem. 78

4.2 Characteristics of neural network rule extraction techniques. 80

4.3 The thermometer encoding procedure for ordinal variables. 81

4.4 The dummy encoding procedure for nominal variables. 82

4.5 The fidelity measure. 89

4.6 Neural network rule extraction results for the continuous data sets. 91

4.7 Fidelity rates of extraction techniques. 91

4.8 Neural network rule extraction results for the discretized data sets. 94

4.9 Fidelity rates of extraction techniques. 94

4.10 The number of columns in the expanded and reduced DTs for the three data sets for the rules and trees extracted by Neurorule and Trepan. 104

5.1 Characteristics of data sets. 123

5.2 Classification accuracy of the evolutionary and neurofuzzy classifiers versus a selection of well-known classification algorithms. 125

6.1 Characteristics of neural network survival analysis methods. 147

6.2 Data set characteristics. 150

6.3 The purpose attribute. 151

6.4 Grouping the purpose attribute for predicting early repayment. . . 154

6.5 Predicting early repayment in first 12 months. 154

6.6 Predicting early repayment 12-24 months. 154

6.7 Grouping the purpose attribute for predicting default. 155

6.8 Predicting default in first 12 months. 158

6.9 Predicting default 12-24 months. 158

6.10 Predicting default in first 12 months on oversampled data set. . . . 159

6.11 Predicting default 12-24 months on oversampled data set. 159

A.1 Training set performance of LS-SVMs on 10 binary data sets. . . . 169

A.2 Optimized hyperparameter values of the LS-SVM. 170

A.3 Optimized hyperparameter values of the SVMs. 170

B.1 Characteristics of the multiclass classification data sets. 171

B.2 The multiclass benchmarking study. 174

C.1 Attributes for the German credit data set. 177

D.1 Attributes for the Bene1 data set. 181

Bibliography

[1] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[3] P.D. Allison. *Survival analysis using the SAS system: a practical guide.* SAS Publishing, Cary, NC, U.S., 1995.

[4] H. Almuallim and T.G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI'91)*, volume 2, pages 547–552, Anaheim, California, 1991. AAAI Press.

[5] R. Andrews, J. Diederich, and A.B. Tickle. A survey and critique of techniques for extracting rules from trained neural networks. *Knowledge Based Systems*, 8(6):373–389, 1995.

[6] R. Andrews and S. Geva. Inserting and extracting knowledge from constrained error back propagation networks. In *Proceedings of the Sixth Australian Conference on Neural Networks*, Sydney, Australia, 1995.

[7] G. Arminger, D. Enache, and T. Bonne. Analyzing credit risk data: A comparison of logistic discrimination, classification tree analysis and feedforward networks. *Computational Statistics*, 12(2):293–310, 1997.

[8] B. Baesens, M. Egmont-Petersen, R. Castelo, and J. Vanthienen. Learning Bayesian network classifiers using Markov Chain Monte Carlo search. In *Proceedings of the Sixteenth International Conference on Pattern Recognition (ICPR)*, pages 49–52, Québec, Canada, 2002. IEEE Computer Society.

[9] B. Baesens, C. Mues, R. Setiono, M. De Backer, and J. Vanthienen. Building intelligent credit scoring systems using decision tables. In *Proceedings of the Fifth International Conference on Enterprise Information Systems (ICEIS'2003)*, pages 19–25, Angers, France, 2003.

[10] B. Baesens, R. Setiono, V. De Lille, S. Viaene, and J. Vanthienen. Neural network rule extraction for credit scoring. In *Proceedings of the Pacific Asian Conference on Intelligent Systems (PAIS)*, pages 128–132, Seoul, Korea, 2001.

[11] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3):312–329, 2003.

[12] B. Baesens, T. Van Gestel, M. Stepanova, and J. Vanthienen. Neural network survival analysis for personal loan data. In *Proceedings of the Eighth Conference on Credit Scoring and Credit Control (CSCCVII'2003)*, Edinburgh, Scotland, 2003.

[13] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen. Benchmarking state of the art classification algorithms for credit scoring. *Journal of the Operational Research Society*, 54(6):627–635, 2003.

[14] B. Baesens, G. Verstraeten, D. Van den Poel, M. Egmont-Petersen, P. Van Kenhove, and J. Vanthienen. Bayesian network classifiers for identifying the slope of the customer lifecycle of long-life customers. *European Journal of Operational Research*, 2003. forthcoming.

[15] B. Baesens, S. Viaene, D. Van den Poel, J. Vanthienen, and G. Dedene. Using Bayesian neural networks for repeat purchase modelling in direct marketing. *European Journal of Operational Research*, 138(1):191–211, 2002.

[16] B. Baesens, S. Viaene, T. Van Gestel, J.A.K. Suykens, G. Dedene, B. De Moor, and J. Vanthienen. An initial approach to wrapped input selection using least squares support vector machine classifiers: Some empirical results. In *Proceedings of the Twelfth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, pages 69–76, Kaatsheuvel, The Netherlands, 2000.

[17] B. Baesens, S. Viaene, T. Van Gestel, J.A.K. Suykens, G. Dedene, B. De Moor, and J. Vanthienen. An empirical assessment of kernel type performance for least squares support vector machine classifiers. In *Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies (KES)*, pages 313–316, Brighton, UK, 2000.

[18] B. Baesens, S. Viaene, J. Vanthienen, and G. Dedene. Wrapped feature selection by means of guided neural network optimisation. In A. Sanfeliu, J.J. Villanueva, M.Vanrell, R. Alquezar, A.K. Jain, and J. Kittler, editors, *Proceedings of The Fifteenth International Conference on Pattern Recognition*, pages 113–116, Barcelona, Spain, 2000. IEEE Computer Society.

[19] B. Bakker, H.J. Kappen, and T.M. Heskes. Improving cox survival analysis with a neural-bayesian approach. *Statistics in Medicine*, 2003. forthcoming.

[20] J. Banasik, J.N. Crook, and L. Thomas. Does scoring a subpopulation make a difference? *International Review of Retail, Distribution and Consumer Research*, 6:180–195, 1996.

[21] J. Banasik, J.N. Crook, and L.C. Thomas. Not if but when will borrowers default. *Journal of the Operational Research Society*, 50:1185–1190, 1999.

[22] J. Banasik, J.N. Crook, and L.C. Thomas. Sample selection bias in credit scoring models. In *Proceedings of the Seventh Conference on Credit Scoring and Credit Control (CSCCVII'2001)*, Edinburgh, Scotland, 2001.

[23] R. Battiti. First- and second-order methods for learning: between steepest descent and Newton's method. *Neural Computation*, 44:141–166, 1992.

[24] S. Bedingfield and K.A. Smith. Evolutionary rule generation and its application to credit scoring. In L. Reznik and V. Kreinovich, editors, *Soft Computing in Measurement and Information Acquisition*, Heidelberg, 2001. Physica-Verlag.

[25] E. Biganzoli, P. Boracchi, L. Mariani, and E. Marubini. Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach. *Statistics in Medicine*, 17:1169–1186, 1998.

[26] C.M. Bishop. *Neural networks for pattern recognition.* Oxford University Press, 1995.

[27] C.L. Blake and C.J. Merz. UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/mlrepository.html]. *Irvine, CA: University of California, Dept. of Information and Computer Science*, 1998.

[28] B.V. Bonnlander. *Nonparametric selection of input variables for connectionist learning.* PhD thesis, University of Colorado, Department of Computer Science, 1996.

[29] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, U.S., 1992. ACM.

[30] L. Bottaci, P.J. Drew, J.E. Hartley, M.B. Hadfield, R. Farouk, P.W.R. Lee, I.M.C. Macintyre, G.S. Duthie, and J.R.T. Monson. Artificial neural networks applied to outcome prediction for colorectal cancer patients in separate institutions. *The Lancet*, 350:469–472, 1997.

[31] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.

[32] P.S. Bradley and O.L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, San Francisco, California, U.S., 1998. Morgan Kaufmann.

[33] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression trees.* Wadsworth and Brooks, Monterey, CA, 1994.

[34] N.E. Breslow. Covariance analysis of censored survival data. *Biometrics*, 30:89–99, 1974.

[35] J.S. Bridle. *Neuro-computing: algorithms, architectures and applications*, chapter Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. Springer-Verlag, 1989.

[36] T. Bäck. Evolutionary algorithms. *ACM SIGBIO Newsletter*, pages 26–31, 1992.

[37] T. Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, 1996.

[38] T. Bäck and H.P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[39] S.F. Brown, A. Branford, and W. Moran. On the use of artificial neural networks for the analysis of survival data. *IEEE Transactions on Neural Networks*, 8:1071–1077, 1997.

[40] H.B. Burke, P.H. Goodman, D.B. Rosen, D.E. Henson, J.N. Weinstein, F.E. Harrell, J.R. Marks, D.P. Winchester, and D.G. Bostwick. Artificial neural networks improve the accuracy of cancer survival prediction. *Cancer*, 79(4):857–862, 1997.

[41] N. Capon. Credit scoring systems: A critical analysis. *Journal of Marketing*, 46:82–91, 1982.

[42] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen. Fuzzy Artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713, 1992.

[43] G.C. Cawley. Matlab support vector machine toolbox (v054.β). [http://theoval.sys.uea.ac.uk/∼gcc/svm/toolbox], University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K.

[44] K.C. Chang, R. Fung, A. Lucas, R. Oliver, and N. Shikaloff. Bayesian networks applied to credit scoring. *IMA Journal of Mathematics Applied in Business and Industry*, 11:1–18, 2000.

[45] P.K. Chintagunta. Variety seeking, purchase timing, and the lightning bolt brand choice model. *Management Science*, 45(4):486–498, 1999.

[46] C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[47] O. Cordón, M.J. del Jesus, and F. Herrera. Genetic learning of fuzzy rule-based classification systems cooperating with fuzzy reasoning methods. *International Journal of Intelligent Systems*, 13(10-11):1025–1053, November 1998.

[48] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases.* Advances in Fuzzy Systems. World Scientific, Singapore, 2001.

[49] D.R. Cox and D. Oakes. *Analysis of survival data.* Chapman and Hall, London, U.K., 1984.

[50] M. Craven. *Extracting Comprehensible Models from Trained Neural Networks.* PhD thesis, Department of Computer Sciences, University of Wisconsin-Madison, 1996.

[51] M. Craven and J. Shavlik. Understanding time-series networks: A case study in rule extraction. *International Journal of Neural Systems*, 8(4):373–384, 1997.

[52] M.W. Craven and J.W. Shavlik. Using sampling and queries to extract rules from trained neural networks. In W.W. Cohen and H. Hirsh, editors, *Proceedings of the Eleventh International Conference on Machine Learning (ICML)*, San Francisco, CA, U.S., 1994. Morgan Kaufmann.

[53] M.W. Craven and J.W. Shavlik. Extracting tree-structured representations of trained networks. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 8, pages 24–30, Cambridge, MA, U.S., 1996. MIT Press.

[54] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[55] J.N. Crook. Who is discouraged from applying for credit? *Economics Letters*, 65:165–172, 1999.

[56] R.M. Cyert and G.L. Thompson. Selecting a portfolio of credit risks by markov chains. *The Journal of Business*, 1:34–46, 1968.

[57] M. De Laurentiis and P. Ravdin. Survival analysis of censored data: neural network analysis detection of complex interactions between variables. *Breast Cancer Research and Treatment*, 32:113–118, 1994.

[58] M. De Laurentiis and P.M. Ravdin. A technique for using neural network analysis to perform survival analysis of censored data. *Cancer Letters*, 77:127–138, 1994.

[59] E.R. De Long, D.M. De Long, and D.L. Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, 44:837–845, 1988.

[60] J.E. Dennis Jr. and R.B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* Prentice Hall, New Jersey, 1983.

[61] V.S. Desai, D.G. Conway, J.N. Crook, and G.A. Overstreet Jr. Credit-scoring models in the credit-union environment using neural networks and genetic algorithms. *IMA Journal of Mathematics Applied in Business and Industry*, 8:323–346, 1997.

[62] V.S. Desai, J.N. Crook, and G.A. Overstreet Jr. A comparison of neural networks and linear scoring models in the credit union environment. *European Journal of Operational Research*, 95(1):24–37, 1996.

[63] T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[64] D.D. Dorfmann and E. Alf. Maximum likelihood estimation of parameters of signal detection theory and determination of confidence intervals - rating-method data. *Journal of Mathematical Psychology*, 6:487–496, 1969.

[65] J. Drew, D.R. Mani, A. Betz, and P. Datta. Targeting customers with statistical and data mining techniques. *Journal of Service Research*, 3(3), 2001.

[66] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley, New York, 1973.

[67] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* John Wiley and Sons, second edition, 2001.

[68] D. Durand. Risk elements in consumer installment financing. *Studies in Consumer Installment Financing: Study 8, National Bureau of Economic Research*, 1941.

[69] B. Efron. The efficiency of logistic regression compared to normal discriminant analysis. *Journal of the American Statistical Society*, 70:892–898, 1975.

[70] B. Efron. The efficiency of Cox's likelihood function for censored data. *Journal of the American Statistical Association*, 72:557–565, 1977.

[71] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap.* Chapman and Hall, 1993.

[72] J.P. Egan. *Signal Detection Theory and ROC analysis. Series in Cognition and Perception.* Academic Press, New York, 1975.

[73] M. Egmont-Petersen, A. Feelders, and B. Baesens. Learning Bayesian network classifiers by minimizing the error rate using Markov Chain Monte Carlo search. *Work in Progress*, 2003. submitted.

[74] R.A. Eisenbeis. Pitfalls in the application of discriminant analysis in business, finance and economics. *Journal of Finance*, 32(3):875–900, 1977.

[75] R.A. Eisenbeis. Problems in applying discriminant analysis in credit scoring models. *Journal of Banking and Finance*, 2:205–219, 1978.

[76] S.S. Erenguc and G.J. Koehler. Survey of mathematical programming models and experimental results for linear discriminant analysis. *Managerial and Decision Economics*, 1990(11):215–225, 1990.

[77] B.S. Everitt. *The analysis of contingency tables.* Chapman and Hall, London, 1977.

[78] D. Faraggi, M. LeBlanc, and J. Crowley. Understanding neural networks using regression trees: an application to multiple myeloma survival data. *Statistics in Medicine*, 20(19):2965–2976, 2001.

[79] D. Faraggi and R. Simon. A neural network model for survival data. *Statistics in Medicine*, 14:73–82, 1995.

[80] T. Fawcett. Using rule sets to maximize ROC performance. In *Proceedings of the IEEE International Conference on Data Mining*, San Jose, California, U.S., 2001.

[81] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[82] U.M Fayyad, G.G. Grinstein, and A. Wierse. *Information Visualization in Data Mining and Knowledge Discovery.* Morgan Kaufmann Publishers, 2001.

[83] U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1022–1029, Chambéry, France, 1993. Morgan Kaufmann.

[84] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining.* MIT Press, Cambridge, MA, 1996.

[85] A. Feelders. Credit scoring and reject inference with mixture models. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 9:1–8, 2000.

[86] A.J. Feelders, A.J.F. le Loux, and J.W. van 't Zand. Data mining for loan evaluation at ABN Amro: A case study. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pages 106–111, 1995.

[87] R.A. Fisher. *Contributions to Mathematical Statistics*. John Wiley and Sons, New York, shewhart, w.a. edition, 1950.

[88] T.C. Fogarty, N.S. Ireson, and S.A. Battles. Developing rule based systems for credit card applications from data with genetic algorithms. *IMA Journal of Mathematics Applied In Business and Industry*, 4:53–59, 1992.

[89] N. Freed and F. Glover. Simple but powerful goal programming models for discriminant problems. *European Journal of Operational Research*, 7:44–60, 1981.

[90] N. Freed and F. Glover. Resolving certain difficulties and improving classification power of LP discriminant analysis formulations. *Decision Sciences*, 1986(17):589–595, 1986.

[91] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'96)*, pages 148–156, 1996.

[92] J. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84:165–175, 1989.

[93] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.

[94] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[95] H. Frydman, J.G. Kallberg, and D.L. Kao. Testing the adequacy of Markov chains and mover-stayer models as representations of credit behaviour. *Operations Research*, 33:1203–1214, 1985.

[96] J.J. Glen. Classification accuracy in discriminant analysis: a mixed integer programming approach. *Journal of the Operational Research Society*, 52:328–339, 2001.

[97] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[98] F. Gönül and M.Z. Shi. Optimal mailing of catalogs: a new methodology using estimable structural dynamic programming models. *Management Science*, 44(9):1249–1262, 1998.

[99] A. González and R. Pérez. Completeness and consistency conditions for learning fuzzy rules. *Fuzzy Sets and Systems*, 96:37–51, 1998.

[100] A. González and F. Herrera. Multi-stage genetic fuzzy systems based on the iterative rule learning approach. *Mathware & Soft Computing*, 4:233–249, 1997.

[101] W. Greene. Sample selection in credit-scoring models. *Japan and the World Economy*, 10:299–316, 1998.

[102] A. Gupta, S. Park, and S.M. Lam. Generalized analytic rule extraction for feedforward neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 11(6):985–991, 1998.

[103] S.K. Halgamuge and M. Glesner. Neural networks in designing fuzzy systems for real world applications. *Fuzzy Sets and Systems*, 65:1–12, 1994.

[104] D.J. Hand. Reject inference in credit operations. In E.F. Mays, editor, *Credit Risk Modelling: Design and Application.* Glenlake Publishing, 1998.

[105] D.J. Hand and W.E. Henley. Statistical classification methods in consumer risk. *Journal of the Royal Statistical Society, Series A*, 160:523–541, 1997.

[106] D.J. Hand and S.D. Jacka. *Discrimination and Classification.* Wiley, 1981.

[107] D.J. Hand and S.D. Jacka. *Statistics in Finance.* Edward Arnold, 1998.

[108] D.J. Hand, K.J. McConway, and E. Stanghellini. Graphical models of applicants for credit. *IMA Journal of Mathematics Applied In Business and Industry*, 8:143–155, 1997.

[109] D.J. Hand and R.J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.

[110] J.A. Hanley and B.J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 148:839–843, 1983.

[111] J.A. Hanley and B.J. McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148:839–843, 1983.

[112] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference, and Prediction.* Springer, 2001.

[113] B. Hauser. *The Direct Marketing Handbook*, chapter List segmentation, pages 233–247. 1992.

[114] Y. Hayashi, R. Setiono, and K. Yoshida. A comparison between two neural network rule extraction techniques for the diagnosis of hepatobiliary disorders. *Artificial Intelligence in Medicine*, 20:205–216, 2000.

[115] K. Helsen and D.C. Schmittlein. Evidence for the effectiveness of hazard rate models. *Marketing Science*, 11(4):395–414, 1993.

[116] W.E. Henley. *Statistical Aspects of Credit Scoring.* PhD thesis, The Open University, Milton Keynes, UK, 1995.

[117] W.E. Henley and D.J. Hand. Construction of a k-nearest neighbour credit-scoring system. *IMA Journal of Mathematics Applied In Business and Industry*, 8:305–321, 1997.

[118] F. Hoffmann. Boosting a genetic fuzzy classifier. In *Proceedings of the Joint Ninth IFSA World Congress and Twentieth NAFIPS International Conference*, pages 1564–1569, Vancouver, Canada, July 2001.

[119] F. Hoffmann. Combining boosting and evolutionary algorithms for learning of fuzzy classification rules. *Fuzzy Sets and Systems*, 2003. forthcoming.

[120] F. Hoffmann, B. Baesens, J. Martens, F. Put, and J. Vanthienen. Comparing a genetic fuzzy and a neurofuzzy classifier for credit scoring. In *Proceedings of the Fifth International FLINS Conference on Computational Intelligent Systems for Applied Research*, pages 355–362, Ghent, Belgium, 2002.

[121] F. Hoffmann, B. Baesens, J. Martens, F. Put, and J. Vanthienen. Comparing a genetic fuzzy and a neurofuzzy classifier for credit scoring. *International Journal of Intelligent Systems*, 17(11):1067–1083, 2002.

[122] R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 3:63–91, 1993.

[123] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[124] H. Ishibuchi, T. Nakashima, and T. Morisawa. Voting in fuzzy rule-based systems for pattern classification problems. *Fuzzy Sets and Systems*, 103:223–238, 1999.

[125] H. Ishibuchi, T. Nakashima, and R. Murata. Three-objective genetics-based machine learning for linguistic rule extraction. *Information Sciences*, 136(1-4):109–133, 2001.

[126] D.C. Jain and N.J. Vilcassim. Investigating household purchase timing decisions: a conditional hazard function approach. *Marketing Science*, 10(1):1–23, 1991.

[127] J.S.R. Jang. Anfis: Adaptive network based fuzzy inference systems. *IEEE Transactions on Systems, Man and Cybernetics*, 23:665–685, 1993.

[128] D.N. Joannes. Reject inference applied to logistic regression for credit scoring. *IMA Journal of Mathematics Applied in Business and Industry*, 5:35–43, 1993.

[129] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In H. Hirsh and W. Cohen, editors, *Proceedings of the Eleventh International Conference on Machine Learning (ICML)*, pages 121–129, San Francisco, 1994. Morgan Kaufmann.

[130] G.H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 338–345, Montreal, Québec, Canada, 1995. Morgan Kaufmann.

[131] I.T. Jollife. *Principal Component Analysis.* Springer-Verlag, New York, 1986.

[132] L. Junco and L. Sanchez. Using the Adaboost algorithm to induce fuzzy rules in classification problems. In *Proceedings of the Spanish Conference for Fuzzy Logic and Technologies (ESTYLF 2000)*, pages 297–301, Sevilla, Spain, September 2000.

[133] J.D. Kalbfleisch and R.L. Prentice. *The statistical analysis of failure time data.* Wiley, New York, 1980.

[134] E.L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53:457–481, 1958.

[135] M.G. Kelly. *Tackling Change and Uncertainty in Credit Scoring.* PhD thesis, The Open University, Milton Keynes, UK, 1998.

[136] M.G. Kelly and D.J. Hand. Credit scoring with uncertain class definitions. *IMA Journal of Mathematics Applied in Business and Industry*, 10:331–345, 1999.

[137] R. Kerber. Chimerge: Discretization of numeric attributes. In *Proceedings of the Ninth National Conference on AI*, pages 123–128, San Jose, CA, U.S., 1992. AAAI Press, The MIT Press.

[138] K. Kira and L. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, pages 129–134, San Jose, CA, U.S., 1992. AAAI Press.

[139] R. Kohavi. *Wrappers for performance enhancement and oblivious decision graphs.* PhD thesis, Department of Computer Science, Stanford University, 1995.

[140] P. Kolesar and J.L. Showers. A robust credit screening model using categorical data. *Management Science*, 31:123–133, 1985.

[141] J.B. Jr. Kruskal. On the shortest spanning subtree of a graph and the travelling salesman problem. In *Proceedings of the American Mathematics Society*, volume 7, pages 48–50, 1956.

[142] J.T. Kwok. The evidence framework applied to support vector machines. *IEEE Transactions on Neural Networks*, 10(5):1018–1031, 2000.

[143] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, pages 223–228, San Jose, CA, U.S., 1992. AAAI Press.

[144] P. Lapuerta, S.P. Azen, and L. LaBree. Use of neural networks in predicting the risk of coronary artery disease. *Computers and Biomedical Research*, 28:38–52, 1995.

[145] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society, Series B*, 50:157–224, 1988.

[146] K.J. Leonard. The development of a rule based expert system model for fraud alert in consumer credit. *European Journal of Operational Research*, 80:350–356, 1995.

[147] H. Levene. Contributions to probability and statistics: Essays in honor of Harold Hotelling, eighth edition. Iowa State University Press, I. Olkin et al. eds, pp. 278-292, 1960.

[148] H.G. Li and D.J. Hand. Direct versus indirect credit scoring classifications. *Journal of the Operational Research Society*, 53(6):637–647, 1997.

[149] L.H. Liebman. A Markov decision model for selecting optimal credit control policies. *Management Science*, 18(10):519–525, 1972.

[150] K. Liestøl, P.K. Andersen, and U. Andersen. Survival analysis and neural nets. *Statistics in Medicine*, 13:1189–1200, 1994.

[151] T.S. Lim, W.Y. Loh, and Y.S. Shih. A comparison of prediction accuracy, complexity and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, 2000.

[152] H. Liu and R. Setiono. Chi2: feature selection and discretization of numeric attributes. In *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 388–391, 1995.

[153] H. Liu and S.T. Tan. X2R: a fast rule generator. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 631–635, Vancouver, Canada, 1995. IEEE Press.

[154] D.J.C. MacKay. *Bayesian methods for Adaptive Models*. PhD thesis, Computation and Neural Systems, California Institute of Technology, Pasadena, CA, U.S., 1992.

[155] D.J.C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736, 1992.

[156] L. Magdalena and F. Monasterio. Evolutionary-based learning applied to fuzzy controllers. In *Proceedings of the Fourth IEEE International Conference on Fuzzy Systems and the Second International Fuzzy Engineering Symposium, (FUZZ-IEEE/IFES'95)*, volume 3, pages 1111–1118, March 1995.

[157] R. Malhotra and D.K. Malhotra. Differentiating between good credits and bad credits using neuro-fuzzy systems. *European Journal of Operational Research*, 136:190–211, 2002.

[158] O.L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 1965(13):444–452, 1965.

[159] D.R. Mani, J. Drew, A. Betz, and P. Datta. Statistics and data mining techniques for lifetime value modeling. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 94–103, San Diego, CA, U.S., 1999.

[160] L. Mariani, D. Coradini, E. Biganzoli, P. Boracchi, E. Marubini, S. Pilotti, B. Salvadori, R. Silvestrini, U. Veronesi, R. Zucali, and F. Rilke. Prognostic factors for metachronous contralateral breast cancer: a comparison of the linear Cox regression model and its artificial neural network extension. *Breast Cancer Research and Treatment*, 44:167–178, 1997.

[161] E. Marubini and M.G. Valsecchi. *Analysing Survival Data from Clinical Trials and Observational Studies.* Statistics in Practice. John Wiley and Sons, Chicester, England, 1995.

[162] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K. R. Müller. Fisher discriminant analysis with kernels. In Y.H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Proceedings of the IEEE Neural Networks for Signal Processing (NNSP) Workshop*, pages 41–48, Madison, Wisconsin, U.S., 1999. IEEE.

[163] T.M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[164] J.E. Moody. Note on generalization, regularization, and architecture selection in nonlinear learning systems. In *Proceedings of the First IEEE-SP Workshop on Neural Networks for Signal Processing*, pages 1–10, Los Alamitos, CA, 1991. IEEE Computer Society Press.

[165] C. Mues. *On the Use of Decision Tables and Diagrams in Knowledge Modeling and Verification.* PhD thesis, Department of Applied Economic Sciences, Katholieke Universiteit Leuven, 2002.

[166] N. Murata, S. Yoshizawa, and S.I. Amari. Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, 5(6):865–872, 1994.

[167] P.M. Murphy and M.J. Pazzani. ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees. In *Proceedings of the Eighth International Machine Learning Workshop*, pages 183–187, Evanston, IL, U.S., 1991. Morgan Kaufmann.

[168] I.T. Nabney. *NETLAB: Algorithms for Pattern Recognition.* Springer Verlag, 2001.

[169] B. Narain. Survival analysis and the credit granting decision. In L.C. Thomas, J.N. Crook, and D.B. Edelman, editors, *Credit Scoring and Credit Control*, pages 109–121. Oxford University Press, 1992.

[170] D. Nauck. Data analysis with neuro-fuzzy methods. *Habilitation Thesis, University of Magdeburg*, 2000.

[171] D. Nauck, F. Klawonn, and R. Kruse. *Foundations of Neuro-Fuzzy Systems.* Wiley, New York, 1997.

[172] D. Nauck and R. Kruse. A neuro-fuzzy method to learn fuzzy classification rules from data. *Fuzzy Sets and Systems*, 89:277–288, 1997.

[173] D. Nauck and R. Kruse. Neuro-fuzzy systems for function approximation. *Fuzzy Sets and Systems*, 101:261–271, 1999.

[174] L. Ohno-Machado. Sequential use of neural networks for survival prediction in Aids. *Journal of the American Medical Informatics Association*, 3:170–174, 1996.

[175] L. Ohno-Machado and M.A. Musen. Modular neural networks for medical prognosis: quantifying the benefits of combining neural networks for survival prediction. *Connection Science*, 9(1):71–86, 1997.

[176] J. Pearl. *Probabilistic reasoning in Intelligent Systems: networks for plausible inference.* Morgan Kaufmann, 1988.

[177] K. Pelckmans, J.A.K. Suykens, T. Van Gestel, J. De Brabanter, L. Lukas, B. Hamers, B. De Moor, and J. Vandewalle. LS-SVMlab : a Matlab/C toolbox for least squares support vector machines. Technical Report 02-44, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2002.

[178] S. Piramuthu. Feature selection for financial credit-risk evaluation decisions. *Informs Journal on Computing*, 11(3):258–266, 1999.

[179] S. Piramuthu. Financial credit-risk evaluation with neural and neurofuzzy systems. *European Journal of Operational Research*, 112(2):310–321, 1999.

[180] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning.* MIT Press, 1999.

[181] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, Cambridge, MA, 1999. MIT Press.

[182] R.L. Prentice and L.A. Gloeckler. Regression analysis of grouped survival data with application to breast cancer data. *Biometrics*, 34:57–67, 1978.

[183] S.J. Press and S. Wilson. Choosing between logistic regression and discriminant analysis. *Journal of the American Statistical Association*, 73(364):699–705, 1978.

[184] F. Provost and P. Domingos. Well-trained PETS: Improving probability estimation trees. CDER Working Paper 00-04, Stern School of Business, New York University, New York, U.S., 2000.

[185] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing classifiers. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 445–453, San Francisco, CA, U.S., 1998. Morgan Kaufmann.

[186] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[187] J.R. Quinlan. *C4.5 programs for machine learning.* Morgan Kaufmann, 1993.

[188] P. Rao. *Nonparametric Functional Estimation.* Academic Press, Orlando, 1983.

[189] P.M. Ravdin and G.M. Clark. A practical application of neural network analysis for predicting outcome of individual breast cancer patients. *Breast Cancer Research and Treatment*, 22:285–293, 1992.

[190] R. Reed. Pruning algorithms-a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747, 1993.

[191] A.P.N. Refenes and A.D. Zapranis. Neural network model identification, variable selection and model adequacy. *Journal of Forecasting*, 18:299–332, 1999.

[192] M.D. Richard and R.P. Lippmann. Neural network classifiers estimate Bayesian a-posteriori probabilities. *Neural Computation*, 3(4):461–483, 1991.

[193] B.D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[194] B.D. Ripley and R.M. Ripley. *Artificial Neural Networks: Prospects for Medicine*, chapter Neural networks as statistical methods in survival analysis. Landes Biosciences Publishers, 1998.

[195] R.M. Ripley. *Neural network models for breast cancer prognosis.* PhD thesis, University of Oxford, Department of Engineering Science, U.K., 1998.

[196] E. Rosenberg and A. Gleit. Quantitative methods in credit management: a survey. *Operations Research*, 42:589–613, 1994.

[197] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter Learning internal representations by error propagation, pages 318–362. MIT Press, Reprinted in Anderson and Rosenfeld, Cambridge, MA, U.S., 1986.

[198] M. Saerens, P. Latinne, and C. Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14:21–41, 2001.

[199] L. Santos-Gomez and M.J. Darnel. Empirical evaluation of decision tables for constructing and comprehending expert system rules. *Knowledge Acquisition*, 4:427–444, 1992.

[200] B. Schölkopf, C. Burges, and A. Smola. *Advances in Kernel Methods - Support Vector Learning.* MIT Press, Cambridge, MA, U.S., 1998.

[201] S. Sestito and T. Dillon. *Automated Knowledge Acquisition.* Prentice Hall, 1994.

[202] R. Setiono. A neural network construction algorithm which maximizes the likelihood function. *Connection Science*, 7(2):147–166, 1995.

[203] R. Setiono. Extracting rules from neural networks by pruning and hidden-unit splitting. *Neural Computation*, 9(1):205–225, 1997.

[204] R. Setiono. A penalty function approach for pruning feedforward neural networks. *Neural Computation*, 9(1):185–204, 1997.

[205] R. Setiono. Generating concise and accurate classification rules for breast cancer diagnosis. *Artificial Intelligence in Medicine*, 18(3):205–219, 2000.

[206] R. Setiono and W.K. Leow. FERNN: an algorithm for fast extraction of rules from neural networks. *Applied Intelligence*, 12(1/2):15–25, 2000.

[207] R. Setiono, W.K. Leow, and J.M. Zurada. Extraction of rules from artificial neural networks for nonlinear regression. *IEEE Transactions on Neural Networks*, 13(3):564–577, 2002.

[208] R. Setiono and H. Liu. Symbolic representation of neural networks. *IEEE Computer*, 29(3):71–77, 1996.

[209] R. Setiono and H. Liu. Neural-network feature selector. *IEEE Transactions on Neural Networks*, 8(3):654–662, 1997.

[210] R. Setiono and H. Liu. Neurolinear: from neural networks to oblique decision rules. *Neurocomputing*, 17(1):1–24, 1997.

[211] R. Setiono, J.Y.L. Thong, and C. Yap. Symbolic rule extraction from neural networks: An application to identifying organizations adopting IT. *Information and Management*, 34(2):91–101, 1998.

[212] P. Sewart and J. Whittaker. Fitting graphical models to credit scoring data. *IMA Journal of Mathematics Applied in Business and Industry*, 9:241–266, 1998.

[213] D.J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures.* Chapman and Hall, second edition, 2000.

[214] B.W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, New York, NY, 1986.

[215] M. Smith. *Neural networks for statistical modelling.* Van Nostrand Reinhold, 1993.

[216] A. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 22:211–231, 1998.

[217] G.W. Snedecor and W.G. Cochran. *Statistical Methods.* Iowa State University Press, eighth edition, 1989.

[218] S.A. Solla, E. Levin, and M. Fleisher. Accelerated learning in layered neural networks. *Complex Systems*, 2:625–640, 1988.

[219] E. Stanghellini, K.J. McConway, and D.J. Hand. A chain graph for applicants for bank credit. *Journal of the Royal Statistical Society, Series C*, 48:239–251, 1999.

[220] A. Steenackers and M.J. Goovaerts. A credit scoring model for personal loans. *Insurance: Mathematics and Economics*, 8:31–34, 1989.

[221] M. Stepanova and L.C. Thomas. PHAB scores: proportional hazards analysis behavioural scores. *Journal of the Operational Research Society*, 52(9):1007–1016, 2001.

[222] M. Stepanova and L.C. Thomas. Survival analysis methods for personal loan data. *Operations Research*, 50(2):277–289, 2002.

[223] W.N. Street. A neural network model for prognostic prediction. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 540–546, Madison, Wisconsin, U.S., 1998. Morgan Kaufmann.

[224] J.A.K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle. Least squares support vector machine classifiers: a large scale algorithm. In *Proceedings of the Fourteenth European Conference on Circuits Theory and Design (ECCTD)*, pages 839–842, Stresa, Italy, 1999.

[225] J.A.K. Suykens, L. Lukas, and J. Vandewalle. Sparse least squares support vector machine classifiers. In *Proceedings of the Eighth European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2000.

[226] J.A.K. Suykens and J. Vandewalle. *Nonlinear Modeling: advanced black-box techniques.* Kluwer Academic Publishers, Boston, 1998.

[227] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[228] J.A.K. Suykens and J. Vandewalle. Multiclass least squares support vector machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Washington DC, U.S., 1999.

[229] J.A.K. Suykens and J. Vandewalle. Training multilayer perceptron classifiers based on a modified support vector method. *IEEE Transactions on Neural Networks*, 10:907–912, 1999.

[230] J.A. Swets. ROC analysis applied to the evaluation of medical imaging techniques. *Investigative Radiology*, 14:109–121, 1979.

[231] J.A. Swets and R.M. Pickett. *Evaluation of Diagnostic Systems: Methods from Signal Detection Theory.* Academic Press, New York, 1982.

[232] I.A. Taha and J. Ghosh. Symbolic interpretation of artificial neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 11(3):448–463, 1999.

[233] L. Tarassenko, R. Whitehouse, G. Gasparini, and A. Harris. Neural network prediction of relapse in breast cancer patients. *Neural Computing and Applications*, 4(2):106–114, 1996.

[234] L.C. Thomas. A survey of credit and behavioural scoring: forecasting financial risk of lending to customers. *International Journal of Forecasting*, 16:149–172, 2000.

[235] L.C. Thomas, D.B. Edelman, and J.N. Crook. *Credit Scoring and Its Applications.* SIAM Monographs on Mathematical Modeling and Computation, Philadelphia, U.S., 2002.

[236] S. Thrun. Extracting rules from artificial neural networks with distributed representations. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 7, Cambridge, MA, U.S., 1995. MIT Press.

[237] G.G. Towell and J.W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.

[238] N. Tschichold-Gurman. Generation and improvement of fuzzy classifiers with incremental learning using fuzzy RuleNet. In K. George, J.H. Carrol, E. Deaton, D. Oppenheim, and J. Hightower, editors, *Proceedings of the ACM Symposium on Applied Computing*, pages 466–470, Nashville, NY, U.S., 1995. ACM Press.

[239] D. Van den Poel. *Response Modeling for Database Marketing using Binary Classification.* PhD thesis, K.U.Leuven, 1999.

[240] T. Van Gestel. *From Linear to Kernel Based Methods for Classification, Modelling and Prediction.* PhD thesis, Department of Electrical Engineering, Katholieke Universiteit Leuven, 2002.

[241] T. Van Gestel, B. Baesens, J. Suykens, D.E. Baestaens, J. Vanthienen, and B. De Moor. Bankruptcy prediction with least squares support vector machine classifiers. In *Proceedings of the International Conference on Computational Intelligence for Financial Engineering (CIFEr2003)*, pages 1–8, Hong Kong, 2003.

[242] T. Van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 2003. forthcoming.

[243] T. Van Gestel, J.A.K. Suykens, D.E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, and J. Vandewalle. Predicting financial time series using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks*, 12(4):809–821, 2001.

[244] T. Van Gestel, J.A.K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle. A Bayesian framework for least squares support vector machine classifiers. *Neural Computation*, 15(4):1115–1147, 2002.

[245] J. Vanthienen and E. Dries. Illustration of a decision table tool for specifying and implementing knowledge based systems. *International Journal on Artificial Intelligence Tools*, 3(2):267–288, 1994.

[246] J. Vanthienen and E Dries. A branch and bound algorithm to optimize the representation of tabular decision processes. research report 9602, K.U.Leuven, 1996.

[247] J. Vanthienen, C. Mues, and A. Aerts. An illustration of verification and validation in the modelling phase of KBS development. *Data and Knowledge Engineering*, 27:337–352, 1998.

[248] J. Vanthienen and G. Wets. From decision tables to expert system shells. *Data and Knowledge Engineering*, 13(3):265–282, 1994.

[249] J. Vanthienen, G. Wets, and G. Chen. Incorporating fuzziness in the classical decision table formalism. *Intelligent Systems*, 11:879–891, 1996.

[250] V. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, New-York, U.S., 1995.

[251] V. Vapnik. *Nonlinear Modeling: advanced black-box techniques*, chapter The support vector method of function estimation, pages 55–85. 1998.

[252] V. Vapnik. *Statistical learning theory.* John Wiley, New-York, U.S., 1998.

[253] G. Verstraeten, B. Baesens, D. Van den Poel, M. Egmont-Petersen, P. Van Kenhove, and J. Vanthienen. Targeting long-life customers: Towards a segmented CRM approach. In *Proceedings of the Thirty-First European Marketing Academy Conference (EMAC2002)*, Braga, Portugal, 2002.

[254] S. Viaene, B. Baesens, D. Van den Poel, G. Dedene, and J. Vanthienen. Wrapped input selection using multilayer perceptrons for repeat-purchase modeling in direct marketing. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 10(2):115–126, 2001.

[255] S. Viaene, B. Baesens, T. Van Gestel, J.A.K. Suykens, D. Van den Poel, J. Vanthienen, B. De Moor, and G. Dedene. Knowledge discovery in a direct marketing case using least squares support vector machines. *International Journal of Intelligent Systems*, 16(9):1023–1036, 2001.

[256] S. Viaene, R. Derrig, B. Baesens, and G. Dedene. A comparison of state-of-the-art classification techniques for expert automobile insurance fraud detection. *Journal of Risk And Insurance (Special Issue on Fraud Detection)*, 69(3):433–443, 2002.

[257] V. Vinciotti and D.J. Hand. Scorecard construction with unbalanced class sizes. *Submitted for Publication*, 2002.

[258] G. Weiss and F. Provost. The effect of class distribution on classifier learning. Technical Report ML-TR 43, Department of Computer Science, Rutgers University, New Jersey, U.S., 2001.

[259] D. West. Neural network credit scoring models. *Computers and Operations Research*, 27:1131–1152, 2000.

[260] G. Wets. *Decision Tables in Knowledge-Based Systems: Adding Knowledge Discovery and Fuzzy Concepts to the Decision Table Formalism.* PhD thesis, Department of Applied Economic Sciences, Catholic University of Leuven, Belgium, 1998.

[261] J.C. Wiginton. A note on the comparison of logit and discriminant models of consumer credit behaviour. *Journal of Financial and Quantitative Analysis*, 15:757–770, 1980.

[262] G. Wilkinson. How credit scoring really works? In L.C. Thomas, J.N. Crook, and D.B. Edelman, editors, *Credit Scoring and Credit Control*, pages 141–160, Oxford, 1992. Oxford University Press.

[263] I.H. Witten and E. Frank. *Data mining: practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann, San Francisco, 2000.

[264] M.B. Yobas, J.N. Crook, and P. Ross. Credit scoring using neural and evolutionary techniques. *IMA Journal of Mathematics Applied in Business and Industry*, 11:111–125, 2000.

[265] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[266] H.J. Zimmermann. *Fuzzy set Theory and its Applications.* Kluwer Academic Publishers, Dordrecht, second edition, 1991.

[267] J.M. Zurada. *Introduction to artificial neural systems.* PWS Publishing Company, Boston, 1995.

[268] B. Zuypan, J. Demsar, M. Kattan, R. Beck, and I. Bratko. Machine learning for survival analysis: a case study on recurrence of prostate cancer. *Artificial Intelligence in Medicine*, 20:59–75, 2000.

CURRICULUM VITAE BART BAESENS

Personal Data

Surname	Baesens
First Name	Bart Maurice Marcella
Born	February 27, 1975 (Bruges, Belgium)
Nationality	Belgian
Civil state	Married to Katrien Denys

Address
Bart Baesens
Katholieke Universiteit Leuven
Department of Applied Economic Sciences
Naamsestraat 69
B-3000 Leuven
Belgium
tel. 003216326884
fax. 003216326732
Bart.Baesens@econ.kuleuven.ac.be

Education

Secondary School
Sint-Leocollege (Bruges, Belgium) (1987-1993) Type II secundair onderwijs, Wetenschappelijke A

Higher Education
Commercial Engineer in Management Informatics, K.U.Leuven (1993-1998): Magna Cum Laude

PhD in Applied Economic Sciences
Public defence planned: September 24, 2003

Research Interests

My research deals with the use of data mining and machine learning techniques to tackle management science problems such as customer scoring, bankruptcy prediction, customer lifetime value estimation, churn prediction, ... I hereby consider both the technical theoretical part as well as the practical implementation details. Some key words describing my research are: management information systems; data mining; machine learning; decision support systems; customer scoring; pattern recognition; classification; survival analysis.

Organizational Activities

- I am the secretary and member of the board of directors of the VZW Contactgroep Beleidsinformatica (CBL) since December 2000. CBL is an association of graduated information systems students and has approximately 150 members. It publishes a quarterly journal and organises conferences on state-of-the-art ICT trends.
- Together with the University Center for Statistics at the K.U.Leuven, I organized and planned the course "Data Mining: Searching for Knowledge in Your Data" which took place on January, 14th, 15th, 17th, 21st, 22th, 24th, 2002 and was repeated on September, 3rd, 5th, 9th, 10th, 2002. The courses have on average 20 students (both business people and academicians).
- I am a member of the SAS Belgium and Luxembourg Users Group (BLUES) Committee since October 24th, 2002. This committee consists of a number of academicians and business people which together help to plan SAS events such as the annual BLUES conference.
- Together with my promotor, prof. dr. Jan Vanthienen, we organised the Credit Scoring Workshop at the K.U.Leuven on March 11th, 2003. This workshop was attended by approximately 30 people, both academicians and business people (mainly from financial institutions). I also gave a lecture as part of this workshop.
- I co-chaired a session on classification and data mining for the Fifth International FLINS Conference on Computational Intelligent Systems for Applied Research (FLINS'2002) in Ghent, Belgium.
- I chaired a session on artificial intelligence and decision support systems for the Fifth International Conference on Enterprise Information Systems (ICEIS'2003) in Angers, France.
- I am organising a special session on Data Analysis for the Eighth Online World Conference on Soft Computing in Industrial Applications, September 29th-October 10th, 2003.

Awards and Nominations

When presenting the results and key findings of my research, I have received the following awards and nominations.

- Best Paper nomination Economic and Financial Systems II at the Fifth World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'2001), Orlando, Florida, July, 2001. The main judgement criterion was the quality of the paper.

- Best Paper nomination at the Fourth International Conference on Enterprise Information Systems (ICEIS'2002), Ciudad Real, Spain, April, 2002. The main judgement criterion was the quality of the paper.

- SAS Student Ambassador award at SAS SEUGI, Paris, June, 2002. The main judgement criterion was the quality of the submitted abstract.

- Best Speaker award at the SAS Academic Day in La Tentation, Brussels, May 23rd, 2002. The main judgement criteria were the clarity of the presentation and the speaker's presentation capability.

- Best Speaker award at the SAS Belgium & Luxembourg Users (BLUES) Conference in the Brabanthal, Haasrode, October 24th, 2002. The main judgement criteria were the clarity of the presentation and the speaker's presentation capability.

Ad Hoc Reviewing

I have done ad-hoc reviewing for the following journals.

- Fuzzy Sets and Systems

- Journal of Applied Soft Computing

- Decision Support Systems

- International Journal of Intelligent Systems in Accounting, Finance & Management

- Computational Statistics and Data Analysis

I also regularly write book and article reviews for the ACM Computing Reviews journal.

Interviews

B. Baesens, Reliable credit-risk analysis based on neural networks, Interview in *Banking & Finance*, pp. 96-97, March, 2003

Publications[1]

Journal publications

1. Baesens B., Setiono R., Mues C., Vanthienen J., Using Neural Network Rule Extraction and Decision Tables for Credit-Risk Evaluation, *Management Science*, Volume 49, Number 3, pp. 312-329, March 2003. **SCI 2001 Impact Factor: 1.502**

2. Baesens B., Van Gestel T., Viaene S., Stepanova M., Suykens J.A.K., Vanthienen J., Benchmarking State of the Art Classification Algorithms for Credit Scoring, *Journal of the Operational Research Society*, Volume 54, Number 6, pp. 627-635. **SCI 2001 Impact Factor: 0.438**

3. Baesens B., Verstraeten G., Van den Poel D., Egmont-Petersen M., Van Kenhove P., Vanthienen J., Bayesian Network Classifiers for Identifying the Slope of the Customer Lifecycle of Long-Life Customers, *European Journal of Operational Research*, forthcoming, 2003. **SCI 2001 Impact Factor: 0.494**

4. Van Gestel T., Baesens B., Garcia J., Van Dijcke P., A Support Vector Machine Approach to Credit Scoring, *Bank en Financiewezen*, Volume 2, pp. 73-82, March 2003. **SCI 2001 Impact Factor: -**

5. Van Gestel T., Suykens J.A.K., Baesens B., Viaene S., Vanthienen J., Dedene G., De Moor B., Vandewalle J., Benchmarking Least Squares Support Vector Machine Classifiers, *Machine Learning*, forthcoming, 2003. **SCI 2001 Impact Factor: 1.476**

6. Hoffmann F., Baesens B., Martens J., Put F., Vanthienen J., Comparing a Genetic Fuzzy and a Neurofuzzy Classifier for Credit Scoring, *International Journal of Intelligent Systems*, Volume 17, Issue 11, pp. 1067-1083, 2002. **SCI 2001 Impact Factor: 0.398**

7. Viaene S., Derrig R., Baesens B., Dedene G., A Comparison of State-of-the-Art Classification Techniques for Expert Automobile Insurance Fraud Detection, *Journal of Risk and Insurance*, Special issue on Fraud Detection, Volume 69, Issue 3, pp. 433-443, 2002. **SCI 2001 Impact Factor: 0.196**

8. Baesens B., Viaene S., Van den Poel D., Vanthienen J., Dedene G., Using Bayesian Neural Networks for Repeat Purchase Modelling in Direct Marketing, *European Journal of Operational Research*, Volume 138, Number 1, pp. 191-211, 2002. **SCI 2001 Impact Factor: 0.494**

9. Viaene S., Baesens B., Van den Poel D., Dedene G., Vanthienen J., Wrapped Input Selection using Multilayer Perceptrons for Repeat-Purchase

[1]Authors are always listed in order of contribution.

Modeling in Direct Marketing, *International Journal of Intelligent Systems in Accounting, Finance and Management*, Volume 10, Number 2, pp. 115-126, 2001. **SCI 2001 Impact Factor: -**

10. Viaene S., Baesens B., Van Gestel T., Suykens J.A.K., Van den Poel D., Vanthienen J., De Moor B., Dedene G., Knowledge Discovery in a Direct Marketing Case using Least Squares Support Vector Machines, *International Journal of Intelligent Systems*, Volume 16, Number 9, pp. 1023-1036, 2001. **SCI 2001 Impact Factor: 0.398**

Dutch Journals

1. Baesens B., Mues C., Vanthienen J., Knowledge Discovery in Data: naar performante én begrijpelijke modellen van bedrijfsintelligentie, *Business IN-zicht*, Nummer 12, Maart 2003.
2. Baesens B., Mues C., Vanthienen J., Knowledge Discovery in Data: van academische denkoefening naar bedrijfsrelevante praktijk, *Informatie*, pp. 30-35, Februari, 2003.
3. Souverein M., Baesens B., Viaene S., Vanderbist D., Vanthienen J., Een overzicht van web usage mining en de implicaties voor e-commerce, *Beleidsinformatica Tijdschrift*, Volume 27, Nummer 2, 2001.
4. Baesens B., ORDBMS'en: de object-relationele verzoening, *Beleidsinformatica Tijdschrift*, Volume 24, Nummer 3, 1998.

Contributions to Books

1. Viaene S., Baesens B., Dedene G., Vanthienen J., Van den Poel D., Proof Running Two State-of-the-Art Pattern Recognition Techniques in the Field of Direct Marketing, *Enterprise Information Systems IV*, Piattini M., Filipe J., Braz J. (Eds), Kluwer, 2002.
2. Hoffmann F., Baesens B., Martens J., Put F., Vanthienen J., Comparing a Genetic Fuzzy and a Neurofuzzy Classifier for Credit Scoring, *Proceedings of the Fifth International FLINS Conference on Computational Intelligent Systems for Applied Research (FLINS'2002)*, Ruan D., D'hondt P., Kerre E.E. (Eds), ISBN 981-238-066-3, World Scientific, pp. 355-362, 2002.
3. Baesens B., Setiono R., Mues C., Viaene S., Vanthienen J., Building Credit-Risk Evaluation Expert Systems using Neural Network Rule Extraction and Decision Tables, *New Directions in Software Engineering*, Liber Amicorum M. Verhelst, Vandenbulcke J. and Snoeck M. (Eds.), 160 pp., Leuven University Press, 2001.

Conference Publications

1. Baesens B.,Van Gestel T., Stepanova M., Vanthienen J., Neural Network Survival Analysis for Personal Loan Data, *Proceedings of the Eighth Conference on Credit Scoring and Credit Control (CSCCVII'2003)*, Edinburgh, Scotland, September, 2003.

2. Egmont-Petersen M., Baesens B., Feelders A., Using Bayesian Networks for Estimating the Risk of Default in Credit Scoring, *Proceedings of the International workshop on Computational Management Science, Economics, Finance and Engineering*, Limassol, Cyprus, March 2003.

3. Baesens B., Mues C., Setiono R., De Backer M., Vanthienen J., Building Intelligent Credit Scoring Systems using Decision Tables, *Proceedings of the Fifth International Conference on Enterprise Information Systems (ICEIS'2003)*, Angers, France, pp. 19-25, April 2003.

4. Van Gestel T., Baesens B., Suykens J.A.K., Espinoza M., Baestaens D.E., Vanthienen J., De Moor B., Bankruptcy Prediction with Least Squares Support Vector Machine Classifiers, *Proceedings of the IEEE International Conference on Computational Intelligence for Financial Engineering (CIFEr2003)*, Hong Kong, pp. 1-8, March 2003.

5. Buckinx W., Baesens B., Van den Poel D., Van Kenhove P., Vanthienen J., Using Machine Learning Techniques to Predict Defection of Top Clients, *Proceedings of the Third International Conference on Data Mining Methods and Databases for Engineering, Finance and Other Fields*, Bologna, Italy, pp. 509-517, September 2002.

6. Baesens B., Egmont-Petersen M., Castelo R., Vanthienen J., Learning Bayesian Network Classifiers for Credit Scoring using Markov Chain Monte Carlo Search, *Proceedings of the Sixteenth International Conference on Pattern Recognition (ICPR'2002)*, IEEE Computer Society, Québec, Canada, pp. 49-52, August 2002.

7. Verstraeten G., Baesens B.,Van den Poel D., Egmont-Petersen M., Van Kenhove P., Vanthienen J., Targeting Long-Life Customers: Towards a Segmented CRM Approach, *Proceedings of the Thirty-First European Marketing Academy Conference (EMAC'2002)*, Braga, Portugal, May 2002.

8. Viaene S., Baesens B., Dedene G., Vanthienen J., Van den Poel D., Proof Running Two State-of-the-Art Pattern Recognition Techniques in the Field of Direct Marketing, *Proceedings of the Fourth International Conference on Enterprise Information Systems (ICEIS'2002)*, Ciudad Real, Spain, pp. 446-454, April 2002. **Best paper nomination**

9. Baesens B., Setiono R., Mues C., Viaene S., Vanthienen J., Building Credit-Risk Evaluation Expert Systems using Neural Network Rule Extraction and Decision Tables, *Proceedings of the Twenty Second International Conference on Information Systems (ICIS'2001)*, New Orleans, Louisiana, USA, December, 2001.

10. Baesens B., Setiono R., De Lille V., Viaene S., Vanthienen J., Neural Network Rule Extraction for Credit Scoring, *Proceedings of The Pacific Asian Conference on Intelligent Systems (PAIS'2001)*, Seoul, Korea, pp. 128-132, November, 2001

11. Viaene S., Derrig R., Baesens B., Dedene G., A Comparison of State-of-the-Art Classification Techniques for Expert Automobile Insurance Fraud Detection, *Proceedings of the Fifth International Congress on Insurance: Mathematics and Economics (IME'2001)*, Pennsylvania, USA, July, 2001.

12. Viaene S., Baesens B., Van den Poel D., Vanthienen J., Dedene G., The Bayesian Evidence Framework for Database Marketing Modeling using both RFM and Non-RFM Predictors, *Proceedings of the Fifth World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'2001)*, Orlando, Florida, USA, pp.136-140, July, 2001. **Best paper nomination**

13. Baesens B., Viaene S., Vanthienen J., A Comparative Study of State of the Art Classification Algorithms for Credit Scoring, *Proceedings of the Seventh Conference on Credit Scoring and Credit Control (CSCCVII'2001)*, Edinburgh, Scotland, September, 2001.

14. Baesens B., Viaene S., Van Gestel T., Suykens J.A.K., Dedene G., De Moor B., Vanthienen J., An Initial Approach to Wrapped Input Selection using Least Squares Support Vector Machine Classifiers: Some Empirical Results, *Proceedings of the Twelfth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'00)*, Kaatsheuvel, The Netherlands, pp. 69-76, November, 2000.

15. Baesens B., Viaene S., Vanthienen J., Dedene G., Wrapped Feature Selection by means of Guided Neural Network Optimisation, *Proceedings of the Fifteenth International Conference on Pattern Recognition (ICPR'2000)*, IEEE Computer Society, Barcelona, Spain, pp. 113-116, September, 2000.

16. Viaene S., Baesens B., Van Gestel T., Suykens J.A.K., Van den Poel D., Vanthienen J., De Moor B., Dedene G., Knowledge Discovery using Least Squares Support Vector Machine Classifiers: a Direct Marketing Case, *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2000)*, D.A. Zighed, J. Komorowski and J. Zytkow (Eds.), Lecture Notes in Artificial Intelligence 1910, Springer, Lyon, France, pp. 657-664, September, 2000. **SCI 2000 Impact Factor: 0.253**

17. Baesens B., Viaene S., Van Gestel T., Suykens J.A.K., Dedene G., De Moor B., Vanthienen J., An Empirical assessment of Kernel Type Performance for Least Squares Support Vector Machine Classifiers, *Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies (KES'2000)*, University of Brighton, UK, pp. 313-316, September, 2000.

18. Viaene S., Baesens B., Van den Poel D., Dedene G., Vanthienen J., Wrapped Feature Selection for Binary Classification Bayesian Regularisation Neural Networks: a Database Marketing Application, *Proceedings of the Second International Conference on DATA MINING 2000*, Cambridge University, UK, pp. 353-362, July, 2000.

19. Baesens B., Viaene S., Vanthienen J., Post-Processing of Association Rules, Proceedings of the special workshop on post-processing, *The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'2000)*, Boston, MA, USA, pp. 2-8, August, 2000.

20. Baesens B., Viaene S., Vanthienen J., Post-Processing of Association Rules, *Proceedings of the VIII Seminar on Knowledge Acquisition in Databases*, Turawa, Poland, pp.159-173, May, 2000.

21. Viaene S., Baesens B., Dedene G., Vanthienen J., Vandenbulcke J., Sensitivity Based Pruning of Input Variables by means of Weight Cascaded Retraining, *Proceedings of the Fourth International Conference and Exhibition on the Practical Application of Knowledge Discovery and Data Mining (PADD'2000)*, Manchester, UK, pp.141-159, April, 2000.

Book and Article Reviews

1. Baesens B., 90 days to the data mart, Simon Alan, John Wiley & Sons, Inc., New York, NY, 1998, 338 pp., ISBN 0-471-25194-1, *Computing Reviews*, Volume 39, Number 10, October 1998, pp. 510-511, ACM Press.

2. Baesens B., Interactive data warehousing, Singh Harry, Prentice Hall PTR, Upper Saddle River, NJ, 1998, 481 pp., ISBN 0-13-080371-5, *Computing Reviews*, Volume 40, Number 8, August 1999, pp. 385-386, ACM Press.

3. Baesens B., Exploring data mining implementation, Hirji K., Communications of the ACM, 44(7): 87-93, 2001, *Computing Reviews*, September 2001, ACM Press.

4. Baesens B., A fast algorithm for mining sequential patterns from large databases, Lu L., Longxiang Z., An C., Ning C., Journal of Computer Science and Technology, Volume 16, Number 4, pp. 359-370, *Computing Reviews*, April 2002, ACM Press.

5. Baesens B., Efficient Construction of Regression Trees with Range and Region Splitting, Morimoto Y., Ishii H., Morishita S., Machine Learning, Volume 45, pp. 235-259, *Computing Reviews*, November 2002, ACM Press.

Research reports

1. Baesens B., Verstraeten G.,Van den Poel D., Egmont-Petersen M., Van Kenhove P., Vanthienen J., Bayesian Network Classifiers for Identifying the Slope of the Customer-Lifecycle of Long-Life Customers, Working Paper 02/154, Department of Marketing, Ghent University (Ghent, Belgium), 2002.

2. Van Gestel T., Baesens B., Suykens J.A.K., Baestaens D., Willekens M., Vanthienen J., De Moor B., Bayesian Kernel Based Classification for Financial Distress Detection, Internal Report 02-127, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2002

3. Van Gestel T., Baesens B., Suykens J.A.K., Baestaens D.E., Vanthienen J., De Moor B., Bankruptcy Prediction with Least Squares Support Vector Machine Classifiers, Internal Report 02-112, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2002.

4. Baesens B., Egmont-Petersen M., Castelo R., Vanthienen J., Learning Bayesian network classifiers for credit scoring using Markov Chain Monte Carlo search, Technical report UU-CS-2001-58, Institute of Computer and Information Sciences, Utrecht University (Utrecht, The Netherlands), 2001.

5. Viaene S., Baesens B., Van den Poel D., Vanthienen J., Dedene G., Bayesian Neural Network Learning for Repeat Purchase Modelling in Direct Marketing, Working Paper 01/105, Department of Marketing, Ghent University (Ghent, Belgium), 2001.

6. Viaene S., Baesens B., Van Gestel T., Suykens J.A.K., Van den Poel D., Vanthienen J., De Moor B., Dedene G., Knowledge Discovery in a Direct Marketing Case using Least Squares Support Vector Machines, Working Paper 01/104, Department of Marketing, Ghent University (Ghent, Belgium), 2001.

7. Viaene S., Baesens B., Van den Poel D., Dedene G., Vanthienen J., Wrapped Input Selection using Multilayer Perceptrons for Repeat-Purchase Modeling in Direct Marketing, Working Paper 01/102, Department of Marketing, Ghent University (Ghent, Belgium), 2001.

8. Baesens B., Viaene S., Van Gestel T., Suykens J.A.K., Dedene G., De Moor B., Vanthienen J., An Empirical assessment of Kernel Type Performance for Least Squares Support Vector Machine Classifiers, Internal Report 00-52, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2000.

9. Van Gestel T., Suykens J.A.K., Baesens B., Viaene S., Vanthienen J., Dedene G., De Moor B., Vandewalle J., Benchmarking Least Squares Support Vector Machine Classifiers, Internal Report 00-37, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2000.

10. Viaene S., Baesens B., Van Gestel T., Suykens J.A.K., Van den Poel D., Vanthienen J., De Moor B., Dedene G., Knowledge Discovery using Least Squares Support Vector Machine Classifiers: a Direct Marketing Case, Internal Report 00-33, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2000.

11. Viaene S., Baesens B., Van Gestel T., Suykens J.A.K., Dedene G., De Moor B., Vanthienen J., Least Squares Support Vector Machine Classifiers : An Empirical Evaluation, Internal Report 00-03, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2000.

12. Viaene S., Baesens B., Van den Poel D., Vanthienen J., Dedene G., Bayesian Neural Network Learning for Repeat Purchase Modelling in Direct Marketing, Technical report nr 0114, ETEW, K.U.Leuven (Leuven, Belgium), 2000.

13. Viaene S., Baesens B., Van den Poel D., Dedene G., Vanthienen J., Wrapped Feature Selection for Neural Networks in Direct Marketing, Technical report nr 0019, ETEW, K.U.Leuven (Leuven, Belgium), 2000.

14. Baesens B., Viaene S., Vanthienen J., Post-Processing of Association Rules, Technical report nr 0020, ETEW, K.U.Leuven (Leuven, Belgium), 2000.

15. Baesens B., Viaene S., Van Gestel T., Suykens J.A.K., Dedene G., De Moor B., Vanthienen J., Least Squares Support Vector Machine Classifiers: An Empirical Evaluation, Technical report nr 0003, ETEW, K.U.Leuven (Leuven, Belgium), 2000.

16. Viaene S., Baesens B., Dedene G., Vanthienen J., Vandenbulcke J., Sensitivity Based Pruning of Input Variables by means of Weight Cascaded Retraining, Technical report nr 9954, ETEW, K.U.Leuven (Leuven, Belgium), 1999.

Doctoral dissertations from the Faculty of Economic and Applied Economic Sciences

(from August 1, 1971)

1. GEPTS, Stefaan
Stability and efficiency of resource allocation processes in discrete commodity spaces.
Leuven, KUL, 1971. 86 pp.

2. PEETERS, Theo
Determinanten van de internationale handel in fabrikaten.
Leuven, Acco, 1971. 290 pp.

3. VAN LOOY, Wim
Personeelsopleiding: een onderzoek naar investeringsaspekten van opleiding.
Hasselt, Vereniging voor wetenschappelijk onderzoek in Limburg, 1971. VII, 238 pp.

4. THARAKAN, Mathew
Indian exports to the European community: problems and prospects.
Leuven, Faculty of economics and applied economics, 1972. X,343 pp.

5. HERROELEN, Willy
Heuristische programmatie: methodologische benadering en praktische toepassing op complexe combinatorische problemen.
Leuven, Aurelia scientifica, 1972. X, 367 pp.

6. VANDENBULCKE, Jacques
De studie en de evaluatie van data-organisatiemethodes en data-zoekmethodes.
Leuven, s.n., 1973. 3 V.

7. PENNYCUICK, Roy A.
The economics of the ecological syndrome.
Leuven, Acco, 1973. XII, 177 pp.

8. KAWATA, T. Bualum
Formation du capital d'origine belge, dette publique et stratégie du développement au Zaire.
Leuven, KUL, 1973. V, 342 pp.

9. DONCKELS, Rik
Doelmatige oriëntering van de sectorale subsidiepolitiek in België een theoretisch onderzoek met empirische toetsing.

Leuven, K.U.Leuven, 1974. VII, 156 pp.

10. VERHELST, Maurice
Contribution to the analysis of organizational information systems and their financial benefits.
Leuven, K.U.Leuven, 1974. 2 V.

11. CLEMEUR, Hugo
Enkele verzekeringstechnische vraagstukken in het licht van de nutstheorie.
Leuven, Aurelia scientifica, 1974. 193 pp.

12. HEYVAERT, Edward
De ontwikkeling van de moderne bank- en krediettechniek tijdens de zestiende en zeventiende eeuw in Europa en te Amsterdam in het bijzonder.
Leuven, K.U.Leuven, 1975. 186 pp.

13. VERTONGHEN, Robert
Investeringscriteria voor publieke investeringen: het uitwerken van een operationele theorie met een toepassing op de verkeersinfrastructuur.
Leuven, Acco, 1975. 254 pp.

14. Niet toegekend.

15. VANOVERBEKE, Lieven
Microeconomisch onderzoek van de sectoriële arbeidsmobiliteit.
Leuven, Acco, 1975. 205 pp.

16. DAEMS, Herman
The holding company: essays on financial intermediation, concentration and capital market imperfections in the Belgian economy.
Leuven, K.U.Leuven, 1975. XII, 268 pp.

17. VAN ROMPUY, Eric
Groot-Brittannië en de Europese monetaire integratie: een onderzoek naar de gevolgen van de Britse toetreding op de geplande Europese monetaire unie.
Leuven, Acco, 1975. XIII, 222 pp.

18. MOESEN, Wim
Het beheer van de staatsschuld en de termijnstructuur van de intrestvoeten met een toepassing voor België.
Leuven, Vander, 1975. XVI, 250 pp.

19. LAMBRECHT, Marc
Capacity constrained multi-facility dynamic lot-size problem.
Leuven, KUL, 1976. 165 pp.

20. RAYMAECKERS, Erik
De mens in de onderneming en de theorie van het producenten-gedrag: een bijdrage tot transdisciplinaire analyse.
Leuven, Acco, 1976. XIII, 538 pp.

21. TEJANO, Albert
Econometric and input-output models in development planning: the case of the Philippines.
Leuven, KUL, 1976. XX, 297 pp.

22. MARTENS, Bernard
Prijsbeleid en inflatie met een toepassing op België.
Leuven, KUL, 1977. IV, 253 pp.

23. VERHEIRSTRAETEN, Albert
Geld, krediet en intrest in de Belgische financiële sector.
Leuven, Acco, 1977. XXII, 377 pp.

24. GHEYSSENS, Lieven
International diversification through the government bond market: a risk-return analysis.
Leuven, s.n., 1977. 188 pp.

25. LEFEBVRE, Chris
Boekhoudkundige verwerking en financiële verslaggeving van huurkooptransacties en verkopen op afbetaling bij ondernemingen die aan consumenten verkopen.
Leuven, KUL, 1977. 228 pp.

26. KESENNE, Stefan
Tijdsallocatie en vrijetijdsbesteding: een econometrisch onderzoek.
Leuven, s.n., 1978. 163 pp.

27. VAN HERCK, Gustaaf
Aspecten van optimaal bedrijfsbeleid volgens het marktwaardecriterium: een risico-rendementsanalyse.
Leuven, KUL, 1978. IV, 163 pp.

28. VAN POECK, Andre
World price trends and price and wage development in Belgium: an investigation into the relevance of the Scandinavian model of inflation for Belgium.
Leuven, s.n., 1979. XIV, 260 pp.

29. VOS, Herman
De industriële technologieverwerving in Brazilië: een analyse.

Leuven, s.n., 1978. onregelmatig gepagineerd.

30. DOMBRECHT, Michel
Financial markets, employment and prices in open economies.
Leuven, KUL, 1979. 182 pp.

31. DE PRIL, Nelson
Bijdrage tot de actuariële studie van het bonus-malussysteem.
Brussel, OAB, 1979. 112 pp.

32. CARRIN, Guy
Economic aspects of social security: a public economics approach.
Leuven, KUL, 1979. onregelmatig gepagineerd

33. REGIDOR, Baldomero
An empirical investigation of the distribution of stock-market prices and weak-form efficiency of the Brussels stock exchange.
Leuven, KUL, 1979. 214 pp.

34. DE GROOT, Roger
Ongelijkheden voor stop loss premies gebaseerd op E.T. systemen in het kader van de veralgemeende convexe analyse.
Leuven, KUL, 1979. 155 pp.

35. CEYSSENS, Martin
On the peak load problem in the presence of rationizing by waiting.
Leuven, KUL, 1979. IX, 217 pp.

36. ABDUL RAZK ABDUL
Mixed enterprise in Malaysia: the case study of joint venture between Malysian public corporations and foreign enterprises.
Leuven, KUL, 1979. 324 pp.

37. DE BRUYNE, Guido
Coordination of economic policy: a game-theoretic approach.
Leuven, KUL, 1980. 106 pp.

38. KELLES, Gerard
Demand, supply, price change and trading volume on financial markets of the matching-order type. Vraag, aanbod, koersontwikkeling en omzet op financiële markten van het Europese type.
Leuven, KUL, 1980. 222 pp.

39. VAN EECKHOUDT, Marc
De invloed van de looptijd, de coupon en de verwachte inflatie op het opbrengstver-

loop van vastrentende finaciële activa.
Leuven, KUL, 1980. 294 pp.

40. SERCU, Piet
Mean-variance asset pricing with deviations from purchasing power parity.
Leuven, s.n., 1981. XIV, 273 pp.

41. DEQUAE, Marie-Gemma
Inflatie, belastingsysteem en waarde van de onderneming.
Leuven, KUL, 1981. 436 pp.

42. BRENNAN, John
An empirical investigation of Belgian price regulation by prior notification: 1975 - 1979 - 1982.
Leuven, KUL, 1982. XIII, 386 pp.

43. COLLA, Annie
Een econometrische analyse van ziekenhuiszorgen.
Leuven, KUL, 1982. 319 pp.

44. Niet toegekend.

45. SCHOKKAERT, Eric
Modelling consumer preference formation.
Leuven, KUL, 1982. VIII, 287 pp.

46. DEGADT, Jan
Specificatie van een econometrisch model voor vervuilingsproblemen met proeven van toepassing op de waterverontreiniging in België.
Leuven, s.n., 1982. 2 V.

47. LANJONG, Mohammad Nasir
A study of market efficiency and risk-return relationships in the Malaysian capital market.
s.l., s.n., 1983. XVI, 287 pp.

48. PROOST, Stef
De allocatie van lokale publieke goederen in een economie met een centrale overheid en lokale overheden.
Leuven, s.n., 1983. onregelmatig gepagineerd.

49. VAN HULLE, Cynthia (/08/83)
Shareholders' unanimity and optimal corporate decision making in imperfect capital markets.
s.l., s.n., 1983. 147 pp. + appendix.

50. VAN WOUWE, Martine (2/12/83)
Ordening van risico's met toepassing op de berekening van ultieme ruïnekansen.
Leuven, s.n., 1983. 109 pp.

51. D'ALCANTARA, Gonzague (15/12/83)
SERENA: a macroeconomic sectoral regional and national account econometric model for the Belgian economy.
Leuven, KUL, 1983. 595 pp.

52. D'HAVE, Piet (24/02/84)
De vraag naar geld in België.
Leuven, KUL, 1984. XI, 318 pp.

53. MAES, Ivo (16/03/84)
The contribution of J.R. Hicks to macro-economic and monetary theory.
Leuven, KUL, 1984. V, 224 pp.

54. SUBIANTO, Bambang (13/09/84)
A study of the effects of specific taxes and subsidies on a firms' R&D investment plan.
s.l., s.n., 1984. V, 284 pp.

55. SLEUWAEGEN, Leo (26/10/84)
Location and investment decisions by multinational enterprises in Belgium and Europe.
Leuven, KUL, 1984. XII, 247 pp.

56. GEYSKENS, Erik (27/03/85)
Produktietheorie en dualiteit.
Leuven, s.n., 1985. VII, 392 pp.

57. COLE, Frank (26/06/85)
Some algorithms for geometric programming.
Leuven, KUL, 1985. 166 pp.

58. STANDAERT, Stan (26/09/86)
A study in the economics of repressed consumption.
Leuven, KUL, 1986. X, 380 pp.

59. DELBEKE, Jos (03/11/86)
Trendperioden in de geldhoeveelheid van België 1877-1983: een theoretische en empirische analyse van de "Banking school" hypothese.
Leuven, KUL, 1986. XII, 430 pp.

60. VANTHIENEN, Jan (08/12/86)
Automatiseringsaspecten van de specificatie, constructie en manipulatie van beslissingstabellen.
Leuven, s.n., 1986. XIV, 378 pp.

61. LUYTEN, Robert (30/04/87)
A systems-based approach for multi-echelon production/inventory systems.
s.l., s.n., 1987. 3V.

62. MERCKEN, Roger (27/04/87)
De invloed van de data base benadering op de interne controle.
Leuven, s.n., 1987. XIII, 346 pp.

63. VAN CAYSEELE, Patrick (20/05/87)
Regulation and international innovative activities in the pharmaceutical industry.
s.l., s.n., 1987. XI, 169 pp.

64. FRANCOIS, Pierre (21/09/87)
De empirische relevantie van de independence from irrelevant alternatives. Assumptie indiscrete keuzemodellen.
Leuven, s.n., 1987. IX, 379 pp.

65. DECOSTER, André (23/09/88)
Family size, welfare and public policy.
Leuven, KUL. Faculteit Economische en toegepaste economische wetenschappen, 1988. XIII, 444 pp.

66. HEIJNEN, Bart (09/09/88)
Risicowijziging onder invloed van vrijstellingen en herverzekeringen: een theoretische analyse van optimaliteit en premiebepaling.
Leuven, KUL. Faculteit Economische en toegepaste economische wetenschappen, 1988. onregelmatig gepagineerd.

67. GEEROMS, Hans (14/10/88)
Belastingvermijding. Theoretische analyse van de determinanten van de belastingontduiking en de belastingontwijking met empirische verificaties.
Leuven, s.n., 1988. XIII, 409, 5 pp.

68. PUT, Ferdi (19/12/88)
Introducing dynamic and temporal aspects in a conceptual (database) schema.
Leuven, KUL. Faculteit Economische en toegepaste economische wetenschappen, 1988. XVIII, 415 pp.

69. VAN ROMPUY, Guido (13/01/89)
A supply-side approach to tax reform programs. Theory and empirical evidence

for Belgium.
Leuven, KUL. Faculteit Economische en toegepaste economische wetenschappen, 1989. XVI, 189, 6 pp.

70. PEETERS, Ludo (19/06/89)
Een ruimtelijk evenwichtsmodel van de graanmarkten in de E.G.: empirische specificatie en beleidstoepassingen.
Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1989. XVI, 412 pp.

71. PACOLET, Jozef (10/11/89)
Marktstructuur en operationele efficiëntie in de Belgische financiële sector.
Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1989. XXII, 547 pp.

72. VANDEBROEK, Martina (13/12/89)
Optimalisatie van verzekeringscontracten en premieberekeningsprincipes.
Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1989. 95 pp.

73. WILLEKENS, Francois ()
Determinance of government growth in industrialized countries with applications to Belgium.
Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1990. VI, 332 pp.

74. VEUGELERS, Reinhilde (02/04/90)
Scope decisions of multinational enterprises.
Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1990. V, 221 pp.

75. KESTELOOT, Katrien (18/06/90)
Essays on performance diagnosis and tacit cooperation in international oligopolies.
Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1990. 227 pp.

76. WU, Changqi (23/10/90)
Strategic aspects of oligopolistic vertical integration.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1990. VIII, 222 pp.

77. ZHANG, Zhaoyong (08/07/91)
A disequilibrium model of China's foreign trade behaviour.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1991. XII, 256 pp.

78. DHAENE, Jan (25/11/91)
Verdelingsfuncties, benaderingen en foutengrenzen van stochastische grootheden geassocieerd aan verzekeringspolissen en -portefeuilles.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1991. 146 pp.

79. BAUWELINCKX, Thierry (07/01/92)
Hierarchical credibility techniques.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. 130 pp.

80. DEMEULEMEESTER, Erik (23/3/92)
Optimal algorithms for various classes of multiple resource-constrained project scheduling problems.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. 180 pp.

81. STEENACKERS, Anna (1/10/92)
Risk analysis with the classical actuarial risk model: theoretical extensions and applications to Reinsurance.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. 139 pp.

82. COCKX, Bart (24/09/92)
The minimum income guarantee. Some views from a dynamic perspective.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. XVII, 401 pp.

83. MEYERMANS, Eric (06/11/92)
Econometric allocation systems for the foreign exchange market: Specification, estimation and testing of transmission mechanisms under currency substitution.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. XVIII, 343 pp.

84. CHEN, Guoqing (04/12/92)
Design of fuzzy relational databases based on fuzzy functional dependency.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. 176 pp.

85. CLAEYS, Christel (18/02/93)
Vertical and horizontal category structures in consumer decision making: The nature of product hierarchies and the effect of brand typicality.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 348 pp.

86. CHEN, Shaoxiang (25/03/93)
The optimal monitoring policies for some stochastic and dynamic production processes.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 170 pp.

87. OVERWEG, Dirk (23/04/93)
Approximate parametric analysis and study of cost capacity management of computer configurations.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 270 pp.

88. DEWACHTER, Hans (22/06/93)
Nonlinearities in speculative prices: The existence and persistence of nonlinearity in foreign exchange rates.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 151 pp.

89. LIN, Liangqi (05/07/93)
Economic determinants of voluntary accounting choices for R & D expenditures in Belgium.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 192 pp.

90. DHAENE, Geert (09/07/93)
Encompassing: formulation, properties and testing.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 117 pp.

91. LAGAE, Wim (20/09/93)
Marktconforme verlichting van soevereine buitenlandse schuld door private crediteuren: een neo-institutionele analyse.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 241 pp.

92. VAN DE GAER, Dirk (27/09/93)
Equality of opportunity and investment in human capital.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 172 pp.

93. SCHROYEN, Alfred (28/02/94)
Essays on redistributive taxation when monitoring is costly.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 203 pp. + V.

94. STEURS, Geert (15/07/94)
Spillovers and cooperation in research and development.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 266 pp.

95. BARAS, Johan (15/09/94)
The small sample distribution of the Wald, Lagrange multiplier and likelihood ratio tests for homogeneity and symmetry in demand analysis: a Monte Carlo study.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 169 pp.

96. GAEREMYNCK, Ann (08/09/94)
The use of depreciation in accounting as a signalling device.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 232 pp.

97. BETTENDORF, Leon (22/09/94)
A dynamic applied general equilibrium model for a small open economy.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 149 pp.

98. TEUNEN, Marleen (10/11/94)
Evaluation of interest randomness in actuarial quantities.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 214 pp.

99. VAN OOTEGEM, Luc (17/01/95)
An economic theory of private donations.
Leuven. K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 236 pp.

100. DE SCHEPPER, Ann (20/03/95)
Stochastic interest rates and the probabilistic behaviour of actuarial functions.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 211 pp.

101. LAUWERS, Luc (13/06/95)
Social choice with infinite populations.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 79 pp.

102. WU, Guang (27/06/95)
A systematic approach to object-oriented business modeling.

Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 248 pp.

103. WU, Xueping (21/08/95)
Term structures in the Belgian market: model estimation and pricing error analysis.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 133 pp.

104. PEPERMANS, Guido (30/08/95)
Four essays on retirement from the labor force.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 128 pp.

105. ALGOED, Koen (11/09/95)
Essays on insurance: a view from a dynamic perspective.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 136 pp.

106. DEGRYSE, Hans (10/10/95)
Essays on financial intermediation, product differentiation, and market structure.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 218 pp.

107. MEIR, Jos (05/12/95)
Het strategisch groepsconcept toegepast op de Belgische financiële sector.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 257 pp.

108. WIJAYA, Miryam Lilian (08/01/96)
Voluntary reciprocity as an informal social insurance mechanism: a game theoretic approach.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 124 pp.

109. VANDAELE, Nico (12/02/96)
The impact of lot sizing on queueing delays: multi product, multi machine models.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 243 pp.

110. GIELENS, Geert (27/02/96)
Some essays on discrete time target zones and their tails.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 131 pp.

111. GUILLAUME, Dominique (20/03/96)
Chaos, randomness and order in the foreign exchange markets. Essays on the modelling of the markets.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 171 pp.

112. DEWIT, Gerda (03/06/96)
Essays on export insurance subsidization.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 186 pp.

113. VAN DEN ACKER, Carine (08/07/96)
Belief-function theory and its application to the modeling of uncertainty in financial statement auditing.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 147 pp.

114. IMAM, Mahmood Osman (31/07/96)
Choice of IPO Flotation Methods in Belgium in an Asymmetric Information Framework and Pricing of IPO's in the Long-Run.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 221 pp.

115. NICAISE, Ides (06/09/96)
Poverty and Human Capital.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 209 pp.

116. EYCKMANS, Johan (18/09/97)
On the Incentives of Nations to Join International Environmental Agreements.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1997. XV + 348 pp.

117. CRISOLOGO-MENDOZA, Lorelei (16/10/97)
Essays on Decision Making in Rural Households: a study of three villages in the Cordillera Region of the Philippines.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1997. 256 pp.

118. DE REYCK, Bert (26/01/98)
Scheduling Projects with Generalized Precedence Relations: Exact and Heuristic Procedures.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. XXIV + 337 pp.

119. VANDEMAELE Sigrid (30/04/98)
Determinants of Issue Procedure Choice within the Context of the French IPO Market: Analysis within an Asymmetric Information Framework.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. 241 pp.

120. VERGAUWEN Filip (30/04/98)
Firm Efficiency and Compensation Schemes for the Management of Innovative Activities and Knowledge Transfers.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. VIII + 175 pp.

121. LEEMANS Herlinde (29/05/98)
The Two-Class Two-Server Queueing Model with Nonpreemptive Heterogeneous Priority Structures.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. 211 pp.

122. GEYSKENS Inge (4/09/98)
Trust, Satisfaction, and Equity in Marketing Channel Relationships.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. 202 pp.

123. SWEENEY John (19/10/98)
Why Hold a Job ? The Labour Market Choice of the Low-Skilled.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. 278 pp.

124. GOEDHUYS Micheline (17/03/99)
Industrial Organisation in Developing Countries, Evidence from Cte d'Ivoire.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 251 pp.

125. POELS Geert (16/04/99)
On the Formal Aspects of the Measurement of Object-Oriented Software Specifications.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 507 pp.

126. MAYERES Inge (25/05/99)
The Control of Transport Externalities: A General Equilibrium Analysis.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. XIV + 294 pp.

127. LEMAHIEU Wilfried (5/07/99)
Improved Navigation and Maintenance through an Object-Oriented Approach to Hypermedia Modelling.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 284 pp.

128. VAN PUYENBROECK Tom (8/07/99)
Informational Aspects of Fiscal Federalism.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 192 pp.

129. VAN DEN POEL Dirk (5/08/99)
Response Modeling for Database Marketing Using Binary Classification.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 342 pp.

130. GIELENS Katrijn (27/08/99)
International Entry Decisions in the Retailing Industry: Antecedents and Performance Consequences.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 336 pp.

131. PEETERS Anneleen (16/12/99)
Labour Turnover Costs, Employment and Temporary Work.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 207 pp.

132. VANHOENACKER Jurgen (17/12/99)
Formalizing a Knowledge Management Architecture Meta-Model for Integrated Business Process Management.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 252 pp.

133. NUNES Paulo (20/03/2000)
Contingent Valuation of the Benefits of Natural Areas and its Warmglow Component.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. XXI + 282 pp.

134. VAN DEN CRUYCE Bart (7/04/2000)
Statistische discriminatie van allochtonen op jobmarkten met rigide lonen.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. XXIII + 441 pp.

135. REPKINE Alexandre (15/03/2000)

Industrial restructuring in countries of Central and Eastern Europe: Combining branch-, firm- and product-level data for a better understanding of Enterprises' behaviour during transition towards market economy.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. VI + 147 pp.

136. AKSOY, Yunus (21/06/2000)
Essays on international price rigidities and exchange rates.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. IX + 236 pp.

137. RIYANTO, Yohanes Eko (22/06/2000)
Essays on the internal and external delegation of authority in firms.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. VIII + 280 pp.

138. HUYGHEBAERT, Nancy (20/12/2000)
The Capital Structure of Business Start-ups.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. VIII + 332 pp.

139. FRANCKX Laurent (22/01/2001)
Ambient Inspections and Commitment in Environmental Enforcement.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 VIII + 286 pp.

140. VANDILLE Guy (16/02/2001)
Essays on the Impact of Income Redistribution on Trade.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 VIII + 176 pp.

141. MARQUERING Wessel (27/04/2001)
Modeling and Forecasting Stock Market Returns and Volatility.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 V + 267 pp.

142. FAGGIO Giulia (07/05/2001)
Labor Market Adjustment and Enterprise Behavior in Transition.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 150 pp.

143. GOOS Peter (30/05/2001)
The Optimal Design of Blocked and Split-plot experiments.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 X + 224 pp.

144. LABRO Eva (01/06/2001)
Total Cost of Ownership Supplier Selection based on Activity Based Costing and Mathematical Programming.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 217 pp.

145. VANHOUCKE Mario (07/06/2001)
Exact Algorithms for various Types of Project Scheduling Problems. Nonregular Objectives and time/cost Trade-offs. 316
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 316 pp.

146. BILSEN Valentijn (28/08/2001)
Entrepreneurship and Private Sector Development in Central European Transition Countries.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 XVI + 188 pp.

147. NIJS Vincent (10/08/2001)
Essays on the dynamic Category-level Impact of Price promotions.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001.

148. CHERCHYE Laurens (24/09/2001)
Topics in Non-parametric Production and Efficiency Analysis.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 VII + 169 pp.

149. VAN DENDER Kurt (15/10/2001)
Aspects of Congestion Pricing for Urban Transport.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 VII + 203 pp.

150. CAPEAU Bart (26/10/2001)
In defence of the excess demand approach to poor peasants' economic behaviour. Theory and Empirics of non-recursive agricultural household modelling.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001, XIII + 286 blz.

151. CALTHROP Edward (09/11/2001)
Essays in urban transport economics.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001.

152. VANDER BAUWHEDE Heidi (03/12/2001)
Earnings management in an Non-Anglo-Saxon environment.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001, 408 pp.

153. DE BACKER Koenraad (22/01/2002)
Multinational firms and industry dynamics in host countries : the case of Belgium.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, VII + 165 pp.

154. BOUWEN Jan (08/02/2002)
Transactive memory in operational workgroups. Concept elaboration and case study.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, 319 pp. + appendix 102 pp.

155. VAN DEN BRANDE Inge (13/03/2002)
The psychological contract between employer and employee : a survey among Flemish employees.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, VIII + 470 pp.

156. VEESTRAETEN Dirk (19/04/2002)
Asset Price Dynamics under Announced Policy Switching.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, 176 pp.

157. PEETERS Marc (16/05/2002)
One Dimensional Cutting and Packing : New Problems and Algorithms.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002

158. SKUDELNY Frauke (21/05/2002)
Essays on The Economic Consequences of the European Monetary Union.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002

159. DE WEERDT Joachim (07/06/2002)
Social Networks, Transfers and Insurance in Developing countries.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, VI + 129 pp.

160. TACK Lieven (25/06/2002)
Optimal Run Orders in Design of Experiments.

Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, XXXI + 344 pp.

161. POELMANS Stephan (10/07/2002)
Making Workflow Systems work. An investigation into the Importance of Task-appropriation fit, End-user Support and other Technological Characteristics. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, 237 pp.

162. JANS Raf (26/09/2002)
Capacitated Lot Sizing Problems : New Applications, Formulations and Algorithms.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002.

163. VIAENE Stijn (25/10/2002)
Learning to Detect Fraud from enriched Insurance Claims Data (Context, Theory and Applications).
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, 315 pp.

164. AYALEW Tekabe (08/11/2002)
Inequality and Capital Investment in a Subsistence Economy.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002. V + 148 pp.

165. MUES Christophe (12/11/2002)
On the Use of Decision Tables and Diagrams in Knowledge Modeling and Verification.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002. 222 pp.

166. BROCK Ellen (13/03/2003)
The Impact of International Trade on European Labour Markets.
K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002.

167. VERMEULEN Frederic (29/11/2002)
Essays on the collective Approach to Household Labour Supply.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002. XIV + 203 pp.

168. CLUDTS Stephan (11/12/2002)
Combining participation in decision-making with financial participation : theoretical and empirical perspectives.

Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002. XIV + 247 pp.

169. WARZYNSKI Frederic (09/01/2003)
The dynamic effect of competition on price cost margins and innovation.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

170. VERWIMP Philip (14/01/2003)
Development and genocide in Rwanda ; a political economy analysis of peasants and power under the Habyarimana regime.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

171. BIGANO Andrea (25/02/2003)
Environmental regulation of the electricity sector in a European Market Framework.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003, XX + 310 pp.

172. MAES Konstantijn (24/03/2003)
Modeling the Term Structure of Interest Rates Across Countries.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003, V+246 pp.

173. VINAIMONT Tom (26/02/2003)
The performance of One- versus Two-Factor Models of the Term Structure of Interest Rates.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

174. OOGHE Erwin (15/04/2003)
Essays in multi-dimensional social choice.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003, VIII+108 pp.

175. FORRIER Anneleen (25/04/2003)
Temporary employment, employability and training. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

176. CARDINAELS Eddy (28/04/2003)
The role of cost system accuracy in managerial decision making.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003. 144 pp.

177. DE GOEIJ Peter (02/07/2003)
Modeling Time-Varying Volatility and Interest Rates.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003. VII+225 pp.

178. LEUS Roel (19/09/2003)
The generation of stable project plans. Complexity and exact algorithms.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

179.MARINHEIRO Carlos (23/09/2003)
EMU and fiscal stabilisation policy : the case of small countries.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003

180. BAESENS Bart (24/09/2003)
Developing Intelligent Systems for Credit Scoring using Machine Learning Techniques.
Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003. 264 pp.